CR-152287

FEASIBILITY STUDY
FOR A
NUMERICAL AERODYNAMIC SIMULATION FACILITY

(NASA-CR-152287) FEASIBILITY STUDY FOR A N79-26068
NUMERICAL AERODYNAMIC SIMULATION FACILITY.
VOLUME 1 Final Report (Control Data Corp.,
St. Paul, Minn.) 626 p HC A99/MF A01 Unclas
CSCL 14B G3/09 28381

By: N. R. Lincoln

Contributions by: C. N. Arnold
R. O. Bergman
D. B. Bonstrom
T. W. Brinkman
S-H. J. Chiu
S. S. Green
S. D. Hansen
D. L. Klein
H. E. Krohn
R. P. Prow

MAY 1979

Prepared under Contract No. NAS2-9896

CONTROL DATA CORPORATION
Research and Advanced Design Laboratory
4290 Fernwood Street
St. Paul, Minnesota 55112

for

AMES RESEARCH CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

FEASIBILITY STUDY
FOR A
NUMERICAL AERODYNAMIC SIMULATION FACILITY

Volume I – Final Report

By: N. R. Lincoln

Contributions by: C. N. Arnold
R. O. Bergman
D. B. Bonstrom
T. W. Brinkman
S-H. J. Chiu
S. S. Green
S. D. Hansen
D. L. Klein
H.-E. Krohn
R. P. Prow

Prepared under Contract No. NAS2-9896

CONTROL DATA CORPORATION
Research and Advanced Design Laboratory
4290 Fernwood Street
St. Paul, Minnesota 55112

for

AMES RESEARCH CENTER
NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

PREFACE

This report consists of five volumes and a summary report. The summary gives an overview of the project which is documented in detail in the report. Volume V contains Control Data proprietary information and, as such, is given to a limited distribution (NASA only). The five volumes are as follows:

Volume I - Final Report

Division 1 - General Narrative and Rationale
Division 2 - Implicit Method Description and Code
Division 3 - Explicit Method Description and Code
Division 4 - Weather/Climate Application Study
Division 5 - Technology Survey Update
Division 6 - NASF Reliability-Availability Evaluation
Division 7 - Maintenance Study for the NASF
Division 8 - Maintenance Software Alternatives
Division 9 - Installation Organization/Operation
Division 10 - NASF Physical Requirements Update
Division 11 - System Simulation Summary and Results

Volume II - Hardware Specifications/Descriptions

Division 1 - FMP Functional Specification
Division 2 - FMP Instruction Descriptions
Division 3 - System Hardware Descriptions
Division 4 - Loosely Coupled Network Description

Volume III - FMP Language Specification/User Manual

Volume IV - Simulation Model User Manuals

Volume V - Cost and Schedule Projections
(Limited Distribution)

TABLE OF CONTENTS

PAGE

DIVISION 1

TABLE OF CONTENTS

PAGE

TABLE OF CONTENTS

PAGE

DIVISION 2 THE THREE-DIMENSIONAL AERODYNAMIC IMPLICIT CODE

DIVISION 3 THE THREE-DIMENSIONAL

TABLE OF CONTENTS

PAGE

DIVISION 4

DIVISION 5 TECHNOLOGY SURVEY UPDATE

DIVISION 6 NASF RELIABILITY-AVAILABILITY EVALUATION

TABLE OF CONTENTS

DIVISION 7 MAINTENANCE STUDY FOR THE NUMERICAL AERODYNAMIC SIMULATION FACILITY

DIVISION 8 MAINTENANCE SOFTWARE ALTERNATIVES FOR THE 1980s

TABLE OF CONTENTS

PAGE

DIVISION 9 INSTALLATION ORGANIZATION/OPERATION

DIVISION 10 NASF PHYSICAL REQUIREMENTS UPDATE

DIVISION 11 SYSTEM SIMULATOR SUMMARY AND RESULTS

TABLE OF CONTENTS

DIVISION 1

GENERAL NARRATIVE AND RATIONALE

1.0 INTRODUCTION

The technological advances that seem to create a new breakthrough in high speed computer development each passing day unquestionably excite the scientists whose investigations demand seemingly limitless amounts of computational power. Until recent developments in reliable production of high performance Large Scale Integration (LSI) and automated computer design methodology, such insatiable computational requirements had to be met, mainly, by manufacturers of 'standard product' computers. The goals of such standard product machines were necessarily linked to the business objectives of the producing manufacturer. These objectives, of necessity, have been the result of compromises made between many complex factors -- cost, performance, compatibility, software support, product line integration, and the realities of design, schedule, and manufacturability. On the surface, at least, the production of a 'special purpose' computer could avoid these numerous compromises, and thus achieve performance levels for a narrow range of problem charcteristics substantially in excess of what the standard product machines could yield. This premise is based on the assumption that the special purpose machine and the standard product machine would be built from similar if not identical technologies, and with similar if not identical design techniques.

The reason such an approach has not been truly practical for a manufacturer until recent innovations in design and silicon technology have occurred is simply the high degree of risk involved in such a project. The risks are considerable -- cost overrun, schedule delays, reliability, maintainability, software development lead time, attaining performance objectives being just a few that haunt any prospective vendor of a massive central computer system. The risks to the consumer are equally great; however, a clever consumer can at least make the manufacturer assume the burden of financial risk for the hardware itself with judicious use of contract clauses. Despite the incredible risks, the potential for solving a heretofore unsolvable class of problems on such a computing ensemble may justify the challenge, particularly if the special purpose computing facility is successful, resulting in a clear-cut savings in time and dollars.

A particular class of problems has been identified as offering the potential for great gains in cost and time if the appropriate computing system can be found to house them. This set of problems is the simulation of fluid flow around three-demensional bodies, both in wind tunnel environments and in free space. The application of numerical simulations to this field of endeavor promises to yield economies in aircraft design due to reductions in tunnel tests, model designs and construction, and various flight conditions. In addition, particularly in transonic flow analysis, numerical simulation may produce results that would be obscured in physical tunnel tests. This class of problems also exhibits other computational characteristics such as massive quantities of data required for three-dimensional meshes and extremely heavy arithmetic load for each solution. A large central processing system capable of crunching the Navier-Stokes solution seems to be called for in this case. Such a system must be capable of holding the data associated with these very large problems, achieving a problem solution in a reasonable amount of time (say about 10 minutes), and then ordering the results in a form that can be easily understood.

The question then arises, "Can a specially designed computer system be built which will provide the necessary power for this specific set of problems?". A corollary question is "Will a specially designed computer system for flow modeling yield performance substantially greater (a factor of at least 10 times) than a high-performance standard product available in the same time frame?".

It is this set of questions that has been raised by NASA, and submitted to those manufacturers who profess an interest in extremely high performance computer development. To answer these basic questions and those questions which derive from them, such as "What are the risks and costs of such a project?" NASA had engaged Control Data Corporation to pursue a two year study into the feasibility of construction of a centralized Numerical Aerodynamic Simulation Facility (NASF) in the time frame of 1980-1984. This study has been segmented into three parts -- documented in references 1 and 2, and in this report -- each of which address in increasing detail the characteristics and feasibility of a full scale NASF.

1.1 OBJECTIVES

As understood by Control Data, the ultimate goal of NASA is to create a facility for flow simulations that can cope with the volume of data needed for three-dimensional models and with complex computations necessary for a continually maturing mathematical solution to the flow equations. This goal is achievable to the extent that sufficient computer power is available to provide system throughput which can suffice for effective aircraft design as well as meaningful flow research. NASA-Ames researchers have determined that a maximum allowable compute time for the efficient conduct of aerodynamic design is on the order of ten minutes per full solution. In addition, the production models have been determined to need meshes on the order of 100x100x100 data points. These two qualities immediately circumscribe the memory and arithmetic performance requirements for the computational portion of the NASF. The computer industry's best projections for the period 1980-1984 do not reveal any potential standard products that can achieve sustained performance of even 1/10 of the requisite calculational performance of the NASF, let alone the memory capacity. The overall objective of the NASF studies has been then to determine if and how a specialized computer system might be built in this time frame to meet NASA flow modeling goals. A somewhat invisible objective of these three study efforts has been to test the corporate willingness of candidate manufacturers to engage in this high risk activity. Throughout the remainder of this report, this last "hidden" objective must be kept in mind, since despite the best affirmations of feasibility and success for the NASF, the absence of vendor interest or support will guarantee that the endeavor will never be launched.

Given the overall objectives, the first study period was commenced with the following objectives:

a) Assessment of architectural and technology alternatives to creation of the main computational component of the NASF, the Flow Model Processor (FMP).

b) Instructing NASA personnel in the implications of a) above.

c) Identifying computational characteristics of simulation codes.

d) Establishing at least one hardware model, using realistic technology that could achieve the goals of the NASF.

e) Preliminary risk analysis for the conduct of such a project.

f) Identifying the key software development considerations for such a large scale system.

At the conclusion of the first study it was determined that a computer system could be designed around the characteristics of the Navier-Stokes solutions employed by NASA-Ames researchers. To a minimal extent a machine structure was arrived at that could conceivably be constructed with technologies that should be available in the 1980-1984 period. An extension was then launched to the first study, intended to further refine the FMP architecture and to develop additional information for NASA planners who were then deeply involved in making the NASF a reality. The objectives of this study were thus similar in nature to those of the first period, with the exception that certain aspects were to be scrutinized in much greater detail:

a) Review of technological developments as they might apply to the construction of the FMP.

b) Detailed modeling of the FMP to provide structural simulation of candidate code sequences.

c) Analysis of the three-dimensional flow models as they would perform on the proposed FMP structure.

d) Development of detailed reliability data from more refined knowledge of the FMP design.

e) Specification of the general functional capabilities needed for the software systems for the FMP and their relationship to the NASF in which it is imbedded.

This second study concluded again that a machine of the essential power was buildable and could be made to meet acceptable standards of reliability, availability, and maintainability (RAM). Before such a large scale effort could be launched however, additional material needed to be developed. Hence the institution of the final "study" effort of the NASF project. The overall objective of this effort was to provide additional detail and additional answers to NASA scientists and planners so that they might begin the lengthy and arduous procurement process for such a system. The objectives of this final study in order of original importance were:

a) Derivation of detailed and reliable cost data for every segment of the project.

b) Validation of the FMP design for functionality and performance at a design level more detailed than the previous structural model.

c) Simulation and analysis of the data flow among the major FMP components.

d) Development of total NASF system load analytical techniques to provide for configuration evaluation.

e) Detailed specification of programming languages and operating system structure for the FMP.

f) Examination of two computational models with dissimilar characteristics to the aerodynamic codes, specifically a spectral weather code and a finite-difference weather code, to determine their performance on the special purpose flow model processor.

g) Simulation of the final FMP design in execution of the four identified performance metrics: the 3-D implicit and 3-D explicit Navier-Stokes solutions developed by Ames and the spectral and finite-difference weather models developed by other NASA agencies.

h) Development of probable system loads created by potential users of the NASF when it becomes fully operational.

i) Final update on technological alternatives in the 1980-1984 period for construction of an FMP.

1.2 INTERRELATIONSHIPS OF THE VARIOUS STUDIES

All study efforts under this NASF project since its inception have been cooperative and interactive in form and style as regards the relationship between NASA investigators and Control Data engineers. It is also impossible to discuss any reasonable conclusions in this final report without taking into account the other side of this study "triangle", the efforts of the alternate contractor, Burroughs Corporation, as they pursued the same objectives on behalf of NASA. The three-way interaction of these parties, Ames, Control Data, and Burroughs, has not only served NASA's aims well but, at Control Data it is believed that the final FMP and NASF structures of both vendors have benefited by the competitive emphasis that two parallel approaches has provided. Thus, once the first study was completed and published, the heavy reliability emphasis placed by Control Data was adopted in part by Burroughs' designers. In a similar way, Burroughs' continuing concentration on the problems of data flow and accessing in the Navier-Stokes codes were brought to Control Data's attention and affected many redesign decisions for the FMP.

It can then be seen that each study to date owes not only its objectives to the groundwork laid by the previous studies, but even more importantly, each subsequent study derives much material from the evaluation of the competitive report for the previous effort as well as from extensive critique of each study by NASA-Ames personnel. The result of this is that in many instances major structural changes have been made and remade as each study progressed. In addition, conclusions have been drawn and redrawn in several areas due the interaction mentioned before and the changing perspective that comes with the passage of time. For example, the original choice for a bulk, random access memory (RAM) for the Control Data version of the FMP was designated as "bubble memory", with Charge Coupled Devices (CCD) being given second place in consideration. In the intervening two years of these studies, actual hardware has been constructed, certain componentry has come into production and newer components have reached unexpected cost levels. The result is that the intermediate storage for the Control Data FMP is now conceived as consisting of large scale RAM chips of moderate performance, in place of the CCD or bubble memory originally chosen.

This report cannot completely supplant the material developed in the previous study periods. Instead its contents may be said to selectively replace or update previously reported data or conclusions in addition to providing new material in those areas not covered in prior studies. Thus this report, combined with references 1 and 2 constitute all the material developed by this contractor to assist and support the procurement of an

NASF, as well as providing guiding information to aid NASA in its decision processes about the entire project. Given the dynamic nature of the technological evolution, and more importantly the state of the economic climate that directs major manufacturing decisions, many of the approaches and conclusions reached herein can be said to remain valid for a period of no more than a year. This does not mean that the recommendations and predictions given for, say year 1983 will not prove to be correct. What it does mean is that if there is as much as a one year delay in initiating any of the next steps in the NASF procurement, design and construction, the choices for technology, architecture, and support processor systems might be radically altered to achieve better cost, performance, and reliability levels.

1.3 THIS REPORT AND THE OVERALL PROJECT

To provide as much quantitative assurance as possible that the proposed NASF project is feasible, it has been necessary to develop almost all of the hardware and software components to a relatively high degree of detail. In the case of the design chosen by Control Data, this has meant selecting a technology which is in existence and whose manufacturability and performance have already been proven. Using this technology a detailed architecture was developed and from that a design carried to enough detail that a reliable simulation could be produced for it, and relatively high-confidence component counts projected. In addition the support processing system needed to be sized and costed. What is represented in this report then is a model for a NASF/FMP ensemble using a possible approach to meeting NASF goals. If this specimen system is truly feasible, then it follows that there are other systems equally feasible, and NASA's concern about feasibility is satisfied. The candidate system offered in this report represents one which, at this point in development, Control Data considers the best possibility in terms of performance, reliability, and true buildability with minimum risk.

In no way should this candidate architecture and design become one that is specified for the final NASF, since there are still alternatives to be investigated. It should be emphasized that the structure and design numbers offered in this report are to support the possibility of a successful conclusion to NASA's search for an effective facility. The design and structure included here should be evaluated only in light of determining feasibility of the proposed NASF, and should not necessarily be considered as the candidate architecture for such a project to be compared with other competitive schemes, except where Control Data has called attention to the effects of architectural differences on some problem formulations.

1.4 DEPTH OF STUDY

Since this study, as well as others, is to form the basis of the procurement of the complete NASF it must necessarily provide as much detail as possible to support the many activities required of NASA and the NASF manufacturer. Given the broad scope of this study and limitations on the resources available, it was not entirely possible to pursue all aspects of the study to the same level of detail. The various tasks were thus met in a somewhat dynamically assigned priority order:

a) The need for detailed and exhaustive cost data by NASA in the summer of 1978 to assist the preparation of funding requests became the focal point of most of the project's technical resources during the early period of this study. Control Data attempted, within the tight time constraints, to conduct a cost analysis for production of the NASF similar to those analyses undertaken for its own product families. Though it was desired that a confidence factor of 10% be ascribed to this activity, the brief time availaable for full cost detailing made this goal almost impossible to achieve. Instead each major factor was given a separate confidence factor, with the expectation that as this study progressed some of the costing could be reevaluated and confidence improved. In fact, the degree to which some of the unknowns of the summer of 1978 were understood has not improved substantially, and probably will not until actual software design has been carried to completion. This is due to the fact that the major cost uncertainties revolve around the software implementation and maintenance strategies. A good deal of detailed hardware design had to be completed to provide the performance and cost data for this study. In the area of FMP hardware the confidence in the cost data has improved to where it is thought to be within the variance goal of 10 percent, for the most part.

b) Hardware redesign of the FMP was a continuing operation during this study period in response to criticisms from Ames, new aspects of the flow codes that were revealed, and the necessity for improving the performance of the FMP on the weather codes. A greater design concern was the reduction of component counts to improve the cost and reliability of the FMP. This led to the reduction of the number of vector pipelines to 5 (4 active and one spare) using a technological "trick" to double the processing bandwidth of the resulting pipelines.

c) Development of the FMP simulator as a reliable and useful tool for measuring code execution, was a continuing task as the changing characteristics of the machine design had to be injected into the simulator, and operational use of the simulator revealed diagnostic and analytical aids that needed to be added.

d) Development of the NASF system model simulator began in mid 1978 and, as more has become known about the probable environment of the Ames NASF, this simulator has become more important as an evaluative tool for both CDC and NASA reseachers.

e) Development of an FMP programming lanugage which was acceptable to potential NASF users, compiler writers and language standards specialists became an interactive exercise with many alternatives weighed, rejected, or criticized. The final outcome of this effort is given in the section on language analysis and design.

f) Encoding of the implicit 3-D flow model in this language was done to demonstrate the language and how it would be mapped into machine instructions for the FMP.

g) Encoding of portions of the explicit code was done to illustrate the operation of the FMP on code sequences not necessarily similar in computational characteristics to the implicit code.

h) Analysis of the mathematical and computational characteristics of the weather codes was done to determine what the effect of the FMP architecture would be on those models.

i) Operating system software for the FMP and the full NASF system was examined and functional characteristics defined for those components not already available in the standard software that will be available on the front-end machines (support processing systems).

j) A study of the reliability, availability, and maintainability of the FMP was conducted by Control Data Supercomputer Operations reliability specialists.

k) A review of previous technological projections and recommendations was conducted to provide update information on what is realistically available to NASF implementers in the 1980-1984 timeframe.

l) The cost data provided NASA in 1978 was reviewed and updated wherever possible with more recent projections.

A number of activities were not carried out to the extent desired at the outset of this study effort. In some cases resource and time limitations dictated this deficiency, in others changing priorities or interests led to truncating a particular study effort. Some examples of this are:

a) A full-fledged and detailed specification of the FMP operating systems was not produced. The vestigial nature of this operating system (described in reference 2) truly eliminates the need for extensive operating system functions, however the placement of some functions (data editing and analysis, for example) has not yet been decided for the NASF, and thus uncertainty remains as to the need for certain functions in the FMP.

b) A full coding and simulation of the weather models was not done by Control Data. Some portions of the spectral model were coded into FMP FORTRAN and results estimated. In addition, some investigation of the finite difference was done. Discussion of these activities is presented in Division 4.

c) A full coding and simulation of the explicit 3-D flow model was not done. Instead portions of this code which had characteristics dissimilar to the implict code were vectorized in the most straightforward manner possible. This effort was conducted to resolve two questions.

    1) If the FMP is designed primarily to be efficient for the implicit code, what is the degradation in performance to be expected of codes with different computational behavior, such as the explicit code?

    2) What level of performance is achievable by a "first attempt" at utilizing the FMP on the part of new FMP programmers?

2.0 FMP DESIGN

2.1 HARDWARE

The FMP that is described here is the result of an evolution in thinking and implementation since the first attempt to arrive at a sufficient machine structure for the flow model solutions in the first study period of this project. It is, admittedly, based on the processing concepts that have emerged from the Control Data STAR-100 and CYBER 200 computer systems. To improve the project's chances for success it has been alleged from the outset that a good deal of the design, implementation, and software development must be grounded on existing work, or work in progress; the risk of beginning literally "from scratch" on an effort of this magnitude is too great to tempt rational developers into making the attempt. The basic principles for creation of the Control Data Flow Model Processor are then:

a) A massive, centralized memory system which serves as the coordinating medium for data transfer and processng control with sufficient porting to be provided in this memory for a multiplicity of concurrent processes to be carried out.

b) The maximization of functional parallelism which employs concurrency along functional lines rather than providing a multitude of concurrent but identical functional elements.

c) The minimization of the number of identical parallel processing elements through the use of the most aggressive technologies available. It is claimed that two processing elements operating at a clock cycle of two nanoseconds provides superior control, interconnection, and reliability characteristics to an ensemble of forty processors operating at a 40 nanosecond clock cycle, although on the surface both would seem to yield an effective processing rate of one step every nanosecond. (Appendix A provides additional information on clock rates as a measure of performance.)

d) A high bandwidth and multi-access I/O connection to all other processors and storage media in the system, to provide multipathing for system availability as well as for performance reasons.

e) The employment of a FMP-type processor as a computational engine only, leaving all tasks other than the mathematical solution of flow equations to other, conventional processors attached to the system.

Given these desirable principles, a hardware design for the FMP can be completed within the constraints of technology availability, reliability, and maintainablity considerations, and tradeoffs involving physical dimensions, power, cooling, and interconnection limitations. The significance of these tradeoff considerations will now be examined for each major component of the FMP discussing the rationale behind the design presented in detail in the FMP functional and instruction specifications which can be found in Volume II of this report.

2.1.1 MEMORY SYSTEM

Far and away the most important part of the FMP is the memory system, both in impact on the entire design and in cost for the entire machine. Memory capacity is dictated by the requirements of current and projected production problems, and by the predicted needs of a class of research problems that may employ the FMP. If the nominal production problem is based on metric dimensions of 100x100x100 elements, then the implicit code in its present form will require 9 million 64-bit words to retain just the flow variables. Another 5 million words are needed for temporary results generated by each sweep required of the "independent sweep method" of flow code solution. Another couple of million words will be needed to hold locally temporary vectors throughout the various subroutines in the flow codes. The nominal space requirements can then be roughly guaged at 16 million words.

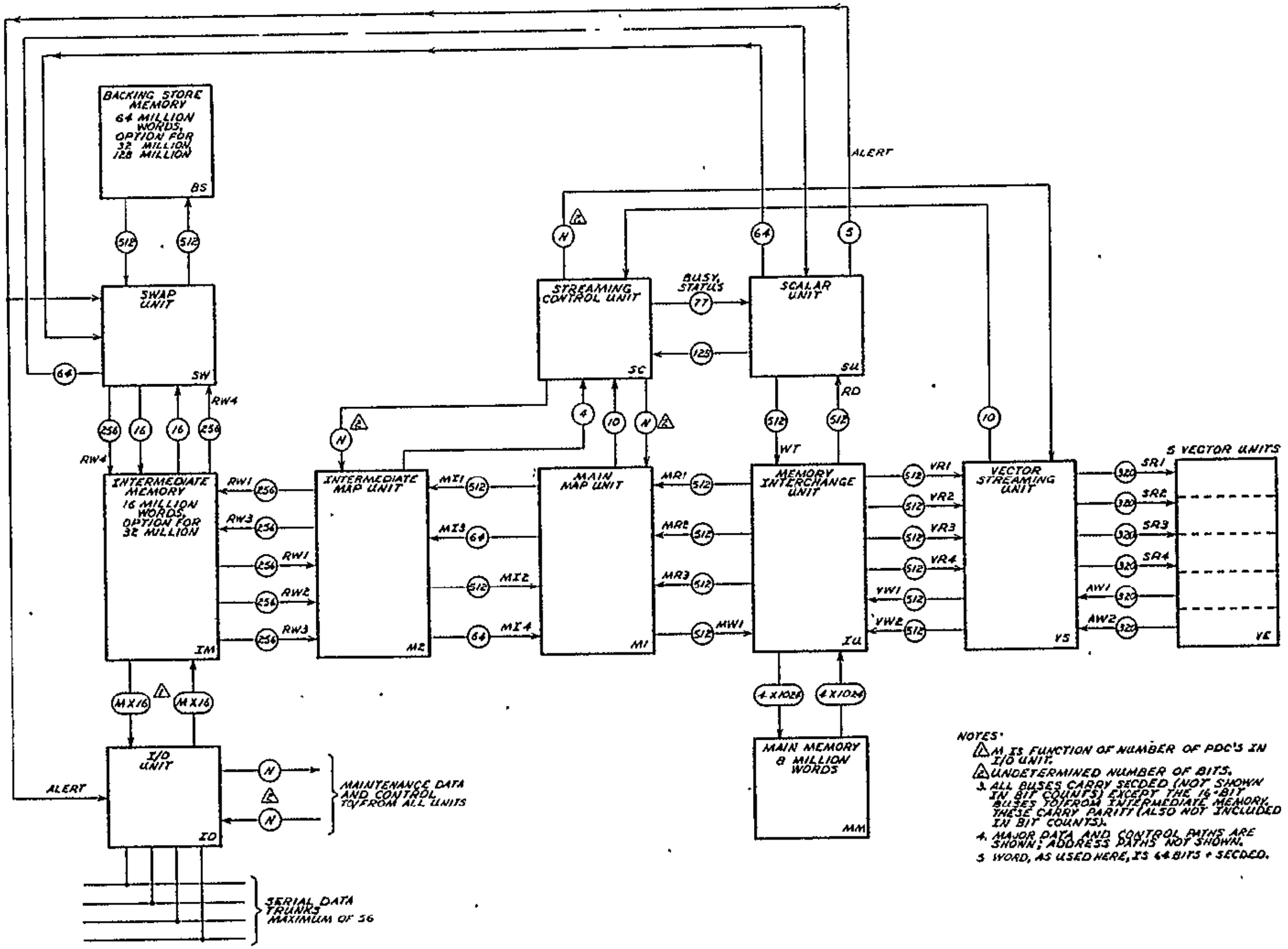
To make the FMP work most efficiently the capability is needed to "stage-in" or "roll-in" one job whilst another is in process so that no time is lost while transferring all the data to be used in and out of the FMP. A buffer space of from 9 to 16 million words seems to be indicated by this strategy.

The term 'roll-in/roll-out' is derived from the Control Data CYBER 70/170 scheme for memory management when a multiplicity of jobs are contending for the CPU. The term usually referred to the act of moving a job's entire CPU memory space onto disk to make room for aother job. It was usually performed by hand on the CDC 6600, and invoked only when the job in memory had a probability of spending a protracted amount of time in an idle state (while an archived tape was being located by the operator, for example). A small, but vital set of data about the job's status was retained by the operating system so that it could be 'rolled in' at a later time and restarted. In variants of this 'roll-in/roll-out' scheme the job, or portions of it, could be moved to extended memory, rather than to disk, to be restored later. In the FMP, the normal mode of operation for batch jobs will consist of readying a complete image of the job on disk, transferring that image to the Backing Store (including code, data base, and supporting parameters), and when space permits in the Intermediate Memory, rolling in the job from Backing Store to Intermediate Memory. At the completion of a job, its entire image is rolled out to backing storage and thence to disk, under some circumstances leaving the job of further data reduction of the rolled-out data to the SPS, or perhaps another job executed on the FMP itself. The basic ability to perform a roll-in/roll-out operation opens a pandora's box of possibilities for complicating the operating system. Performing checkpoint-restart images, for recovery in the event of a system failure, is one possible use of the roll-in/roll-out facility. Another would be 'interrupt roll-out' where, under special circumstances, the SPS (perhaps at the request of the user) interrupts the present job in the CPU. The job could be rolled-out to backing storage or disk until the SPS either performs an ABORT or CONTINUE function. Note that once these facilities are in place, the incentive to take the next potentially fatal step into time-sharing could become too enticing. It is at this point that the systems developers must exert some degree of discipline on FMP operating system design, so that the FMP doesn't become an abused, general-purpose, time-sharing machine, instead of a special-purpose, batch-oriented computational engine.

Finally, the research problems that are contemplated may require basic CPU-contained data bases of the order of 30 to 100 million elements that must be accessed at speeds higher than can be provided by existing rotating mass storage systems. Thus the apparent resulting requirement is for a production code memory of from 16 to 40 million words with expandability to about 200 million words.

Given the current technological predictions on componentry, it is not possible to construct a single, homogenous memory out of one single technology that would provide this range of memory capacity, and still meet the bandwidth requirements of the concurrent functional elements of the FMP.

The overall block diagram (figure 1) shows the FMP to possess a three-level hierarchy of memory. Each level is designed with a particular set of bandwidth, access time, cost, and component counts commensurate with the volume of data contained. That is, in short, the larger the capacity the slower will be the access time and the lower the bandwidth, in exchange for a significant reduction in cost and failure rate on a per-bit basis. A fourth "invisible" level of memory exists which consists of extremely high performance components (effective access times on the order of 3-8 nanoseconds) which are used as register files and high speed buffers in the internal design of all functional units of the FMP.



NOTES:

1. M IS FUNCTION OF NUMBER OF PDC'S IN I/O UNIT.
2. UNDETERMINED NUMBER OF BITS.
3. ALL BUSES CARRY SECDED (NOT SHOWN IN BIT COUNTS) EXCEPT THE 16-BIT BUSES TO/FROM INTERMEDIATE MEMORY. THESE CARRY PARITY (ALSO NOT INCLUDED IN BIT COUNTS).
4. MAJOR DATA AND CONTROL PATHS ARE SHOWN; ADDRESS PATHS NOT SHOWN.
5. WORD, AS USED HERE, IS 64 BITS + SECDED.

Figure 1. Basic FMP Configuration

2.1.1.1 LEVEL 1 MEMORY

The first, and most crucial memory is that called Main Memory (or LEVEL 1 memory). It is this memory that provides the effective bandwidth to supply operands to all parallel functional units. Not only is bandwidth a consideration but single-element access time must be minimized in this level of memory so that those processes which are necessarily "purely" scalar can be carried out at the maximum rate, and essential "transpose" operations on single elements can be accomplished in minimum tme. This level of memory then contains the most powerful and dense memory technology available. At this time the practical limits prescribe the use of a high speed bipolar Random Access Memory (RAM) part which is organized as a 4096 by 1-bit storage device with an access time of 16-20 nanoseconds.

The sheer cost and number of such devices needed to build a basic million-word unit of high performance memory make it impossible to use this technology uniformly throughout the machine. A reasonable limit, based on parts count reliability, power and cooling requirements, and the physical geometry for such a memory which affects access time, is the construction of 8 million 64-bit words employing this technology. If problems can be done in 32-bit mode, this memory could house 16 million 32-bit elements. The speed of this memory part makes it possible to organize the memory into four sets of eight banks which, when strobed in a systematic way, can deliver data (or accept data for storage) at rates up to 1024 bits per set every CPU clock cycle.

This memory is organized in modules and ports with a 32-bit half-word as the smallest writable segment. Thus the single error correction, double error detection system (SECDED) is organized in a similar manner providing 7 bits of error correction/detection for every 32 bits of data stored in memory.

The minimum configuration of this Main Memory is 2 million words, a size necessary to preserve the banking relationships which support the large bandwidth of this memory system. Address trunks and other controls are provided for possible later technological extensions to memory chips of up to 65K bits. In this case the maximum memory available for LEVEL 1 could be 128 million words. It must be pointed out that such a component is not forseen (at the requisite performance levels) for at least five or six years (well beyond the target time frame). Additionally, such parts will be more expensive than the current componentry, and thus might motivate a search for a more dense, and much less expensive part for the other memory hierarchies.

2.1.1.2. LEVEL 2 MEMORY

Since the maximum practical size of the high performance memory has been limited by engineering fiat to 8 million words, some means must be sought for holding the bulk of the nominal flow model data. There exists at present one memory system composed of medium performance 32K by 1-bit semiconductor devices (two 16K by 1-bit chips per package) which can provide a reasonably high bandwidth and single-element access times of approximately 125 nanoseconds. The projection by technological experts that a 131K by 1-bit device (two 64K by 1-bit chips per package) for this system will be available in the timeframe of FMP construction has been established with high confidence. A medium performance system can thus be configured for data storage of from 8 to 32 million 64-bit words (16 to 64 million 32-bit words) with peak bandwidths on the order of 20 billion bits per second. This memory would be organized on a 64-bit basis with 8 bits of SECDED for every 64-bit word.

Note that such a memory trades off an access speed 4 times slower than the Main Memory and a bandwidth 12 times slower for a parts count reduction of 8 times for the same volume of memory, and probable cost ratio in favor of the LEVEL 2 memory of 4-6 to one.

The very nature of Intermediate Memory (LEVEL 2) implies that the time required is greater to deliver data to other functional elements (Vector or Scalar Units, for example), thus some electronic delays are permissible in transmission lines between the Intermediate Memory and the other memory systems. This means that the LEVEL 2 memory can be engineered into a stand-alone unit which eases expandability (for the range of memory configurations) and improves accessibility for maintenance actions.

It is in LEVEL 2 memory that the bulk of all flow model data will reside for the large production problems. Smaller problems may, in fact, be totally contained in the 8 million-word Main Memory. The remainder of the LEVEL 2 memory will be used to stage other jobs in and out while the current job is in progress.

2.1.1.3 LEVEL 3 MEMORY

To simplify hardware scheduling of input and output and to provide a moderate performance memory for the large research problems, a third level of memory is shown on the block diagram (figure 1). This memory would be limited to block transfers only of 32K elements each. By establishing this limitation several high density, low cost, slow access technologies can be employed. This block transfer characteristic is paticularly useful when considering the employment of charge coupled device (CCD) technology. Although the beginning of a particular block may take several milliseconds to reach the output port of the CCD shift register, this wait can be avoided by starting data transfer at any point in the block with the limitation to always transfer an entire block. At the cost of some counters in the CCD memory system and the Swap Unit to which it is attached, the access time to select a given block can be reduced to near zero.

The LEVEL 3 memory supplies or accepts data at an effective rate 32 bits every clock cycle at its single data port. This data moves to/from Intermediate Memory (LEVEL 2) via the Swap Unit. If a 9 million-word job has been set up and held in this memory awaiting execution, it can be rolled in to Intermediate Memory in 18 million clock cycles which is approximately 288 milliseconds, assuming no major conflicts in access to either the LEVEL 2 or LEVEL 3 memory. Since the expected length of execution for the nominal job is on the order of 5 to 10 minutes, there is obviously a large window in which the 288 milliseconds can be expended.

The LEVEL 3 memory can be absent from the FMP configuration if initial installation requirements cannot justify its purchase; in this case however, there will be some degradation in performance where "explicit" input and output are required by the executing code. The transfers to disk cannot be scheduled by the hardware, since a ready-resume LEVEL 3 memory is the normal I/O mechanism for the FMP. Therefore I/O transfers directly to rotating mass storage may involve many "lost revolutions" due to the priorities given the Map and Vector Units for memory access.

A better alternative to initially configuring the LEVEL 3 memory would be to install a minimum CCD memory system of 8 million words, even though this is a smaller capacity than the necessary 32 million words in LEVEL 2 memory (word = 64 bits plus SECDED). Software and hardware would thus operate exactly as it would in the final configuration. The LEVEL 3 memory is designed for a maximum of 256 million words, while current parts projections offer a practical limit of 128 million words for the 1980-1984 construction period.

2.1.1.4 MEMORY TO MEMORY DATA FLOW

Further knowledge of the memory hierarchy might be gained by following the movement of data through the FMP as it might occur for a large production problem. The initial flow field and mesh coordinates will have been stored on rotating mass storage (RMS) prior to initiating data transfer for a particular job.

1) While other jobs are in progress on the FMP, the incoming job's data is moved from RMS to the Backing Store through a buffer in Intermediate Memory. The I/O channel connections to the serial data trunk can provide 50 megabits of data at peak rate, however, the disk transfer rate is 38 megabits per second. With four channels transferring in parallel the rate is 4 x 38 = 152 megabits per second, but, on the average, half the time of each is spent reading and half is

spent writing. Also, latency and gaps on disk reduce the effective rate approximately by 2. Therefore, the nominal 9,000,000 word data base (576,000,000 bits of data) can be moved in 576,000,000/((38,000,000 x 4/2)/2) = 15.16 seconds, if no other memory conflicts or trunk conflicts exist. In reality the time required will be somewhat greater than this, as other demands for Intermediate Memory and Backing Store take priority over the I/O transfer. Sufficient bandwidth is provided to make this factor negligible. Intermediate Memory bandwidth is 21.3 gigabits per second of which about one-fourth (or 5.3 gigabits per second) is available to I/O transfer.

2) Once the data is completely stored in the Backing Store (LEVEL 3 memory) the job is ready for staging into Intermediate Memory for execution. As stated previously, this staging process requires about 288 milliseconds for the entire data base.

3) During job execution the flow model program moves slabs of data from Intermediate Memory to Main Memory using the Map Unit. When processing is completed these slabs are moved back to Intermediate Memory. At the conclusion of the job, the resulting data base is swapped back to the Backing Store (or rolled out of Intermediate Memory). While the job is in execution, the program may call for the dumping of all or portions of the flow variables for output to the user. These I/O operations are normally staged to the Backing Store and then scheduled for transfer to RMS in block transfers at the convenience of the hardware system.

4) The final solution variables and intermediate ones dumped during execution are transferred back to RMS for further editing and evaluation by the support processing system.

Note that the Backing Store is not necessary for execution of the nominal job streams, but in the event that it is absent, the FMP may not be fully utilized. For example, some codes may require a high utilization of Intermediate Memory for data, leaving a small portion of memory available for staging operations. If one-third of the Intermediate Memory is reserved as a staging buffer for the next job, a minimum of 2 times 15.16 seconds will then be required to accomplish the transfer out of Intermediate Memory of a previous job's data and transfer into Intermediate Memory of the next job's data. This is a theoretical minimum of 30.32 seconds. It is possible that some jobs executing in the FMP will finish before the nominal data transfer can be completed and thus the FMP will become idle.

For larger research problems where the data base cannot be completely held within the Intermediate Memory, a segment, or perhaps all, of the data would be held in the Backing Store, and transferred in "slices", "slabs", or "pencils" to the Intermediate Memory and thence dismembered by appropriate map operations. For such cases a Backing Store memory is required.

2.1.2 FUNCTIONAL PARALLELISM

One of the significant outcomes of the initial studies for the NASF was the unanimous conclusion that the computational power of the FMP would have to come from parallel processing techniques as well as from aggressive use of state-of-the-art technology. A major difficulty in the effective employment of parallel processes lies in the means of hardware and software control of such assemblies.

While good progress has been made in programming and management of parallel jobs and, in many cases, parallel tasks in large systems, the focusing of a multitude of identical processors on a single task has not yet reached practical utility in the field of general problem solving, particularly of the type seen in the NASA flow models. One current solution to the programming and control problem has been the evolution of "vector processing" which can perform arithmetic on a data stream (vector) at rates dependent on how many vector operands can be processed in a given clock cycle. This rate is a function of the bandwidth of the "vector unit" and the data ports that feed it. It is a reasonably simple design problem to provide a range of vector unit bandwidths, either by using extremely fast technology in the vector hardware, pipelining the data through the units, or by providing several identical arithmetic elements in a pipeline all of which operate in "lock-step". The designer may also choose any combination or all of the above options to derive arithmetic performance. The key feature of this approach is that the programmer thinks and codes in terms of vectors without attempting to provide separate control to the system for each one of the vector processing elements. If this programming approach is carefully controlled, the user can be made ignorant of the actual number of parallel arithmetic elements actually enagaged at any one moment. The hardware control becomes one of identical operation of such parallel arithmetic units, each on a different portion of the incoming data stream.

An examination of the flow codes shows that arithmetic processing is only a fragment of the total functional processing that must be done, since the data must be organized into vectors where necessary, or must undergo some form of scalar calculation in some cases. As is common in existing computer systems, direct methods exist for the parallel execution of arithmetic functions while data is being transferred concurrently to and from I/O channels; so too, the FMP can perform a variety of data movement activities, all simultaneous with vector arithmetic processing.

This concept of concurrent, asynchronous execution of different functions upon the data in memory is the major programming and hardware control principle of Control Data's proposed Flow Model Processor. The functions which can operate in parallel in this mode are:

1. Input/Output
2. Backing Store swaps
3. Intermediate Memory map operations
4. Main Memory map operations
5. Scalar processing, including management of the instruction stream
6. Vector processing

Input/output execution is similar to that in most other modern day processors. Control of the actual data transfers is handled by the Programmable Device Controllers (PDC) that attach the I/O channels to the network trunk. Requests to the PDC are found in the form of software "messages" which are stored in Intermediate Memory by the Operating System.

Backing Store swaps are handled by the Swap Unit, which behaves much the same as the PDC does in I/O transfers. Requests for swap operations are stored in Intermediate Memory by the Operating System and processed by "firmware" in the Swap Unit.

The previous functions are directed and controlled by software conventions established by developers of the Operating System and firmware. The remaining four parallel functions are hardware implemented processes that are initiated from the single FMP instruction stream. Once initiated, each separate function proceeds somewhat asynchronously until it is complete. Should one function depend on the data being processed by another function, the compiler can establish this relationship and ensure that hardware interlocks will prevent one operation from starting until its predecessor is complete, through the setting of dependency "keys". The coordination of Vector and Map Unit functions is handled by the Streaming Control Unit which resolves dependency conflicts, organizes and distributes the setup data, and initiates the appropriate streaming operation.

Streaming operations involve the processing of sequentially stored (or vector) data. The optimal performance of the memory system of the FMP is achieved when all memory accesses are coordinated and a group of elements can be acquired at each access. The FMP memory design goal is to guarantee that groups of elements can be accessed and transferred to functional units on a regular and continuous basis thus providing an unbroken stream of operands to all attached parallel processors. The Streaming Control Unit (see figure 2) delivers the appropriate setup data to the Map and Vector Units and then initiates the unit's activity. The scheduling of memory requests and the buffering of data are then handled by the specific functional unit (Intermediate Map Unit, Main Map Unit, Vector Streaming Unit).

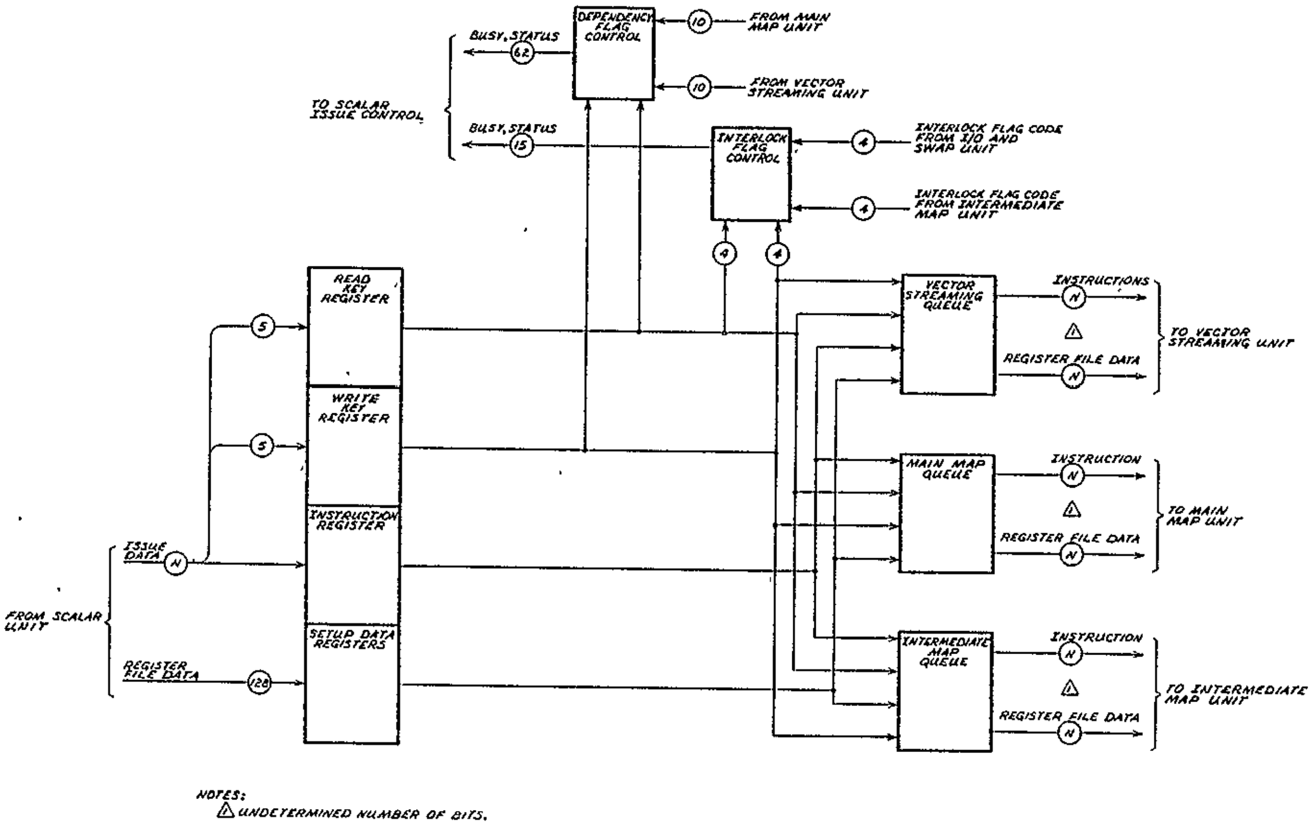


Figure 2. Streaming Control Unit

The Map Units can operate independently with their own memory (Intermediate or Main Memory) and thus proceed concurrently, or they can be linked to perform transfer (map) operations between the two levels of memory. In all instances, the Scalar and Vector Units operate concurrently with any or all other functional units.

2.1.3 THE MAP UNITS

Block diagrams shown in figures 3 and 4 display the organization of the two Map Units. The specification and function of these systems are detailed in Volume II. The major function of the Map Units is to organize data from the original mesh structure into optimal length vectors for processing by the Vector Unit and then to reorganize the results into other mesh structures. The various transformations that provide the functions to be called mapping operations are described below.

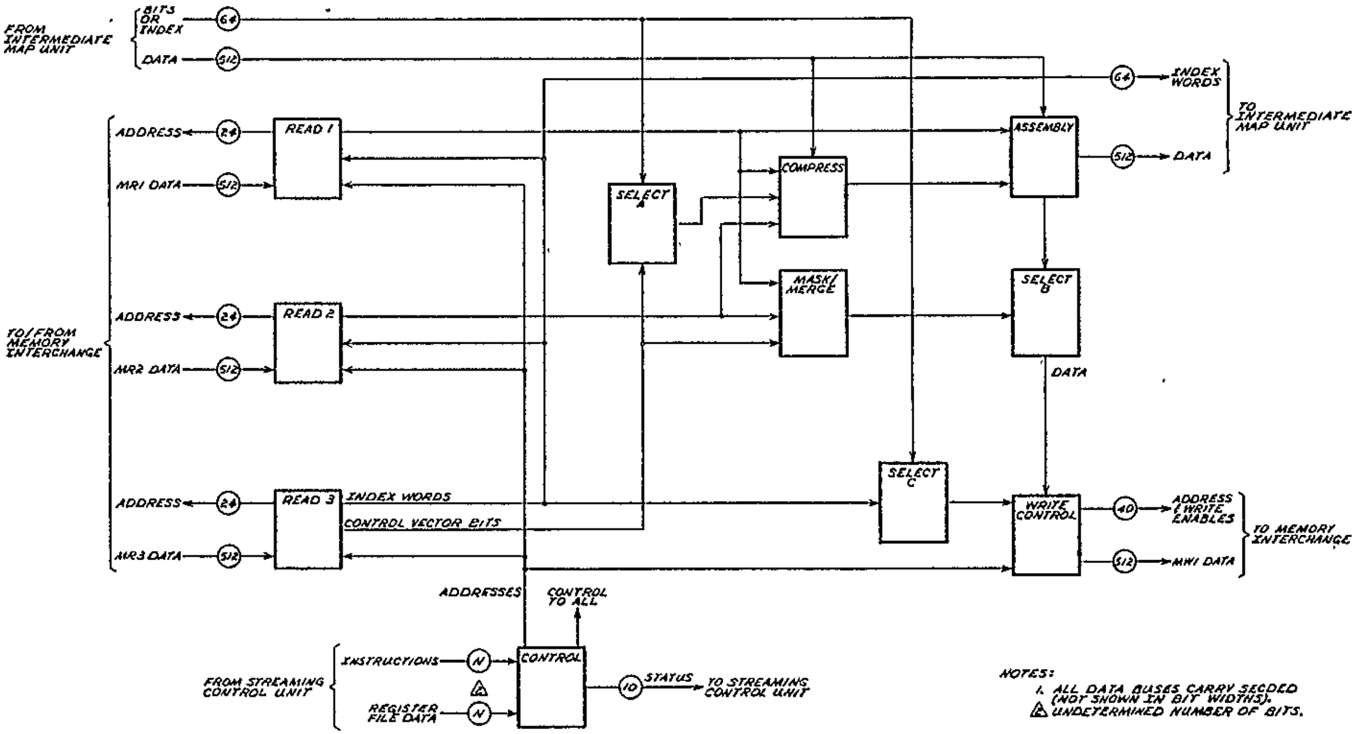
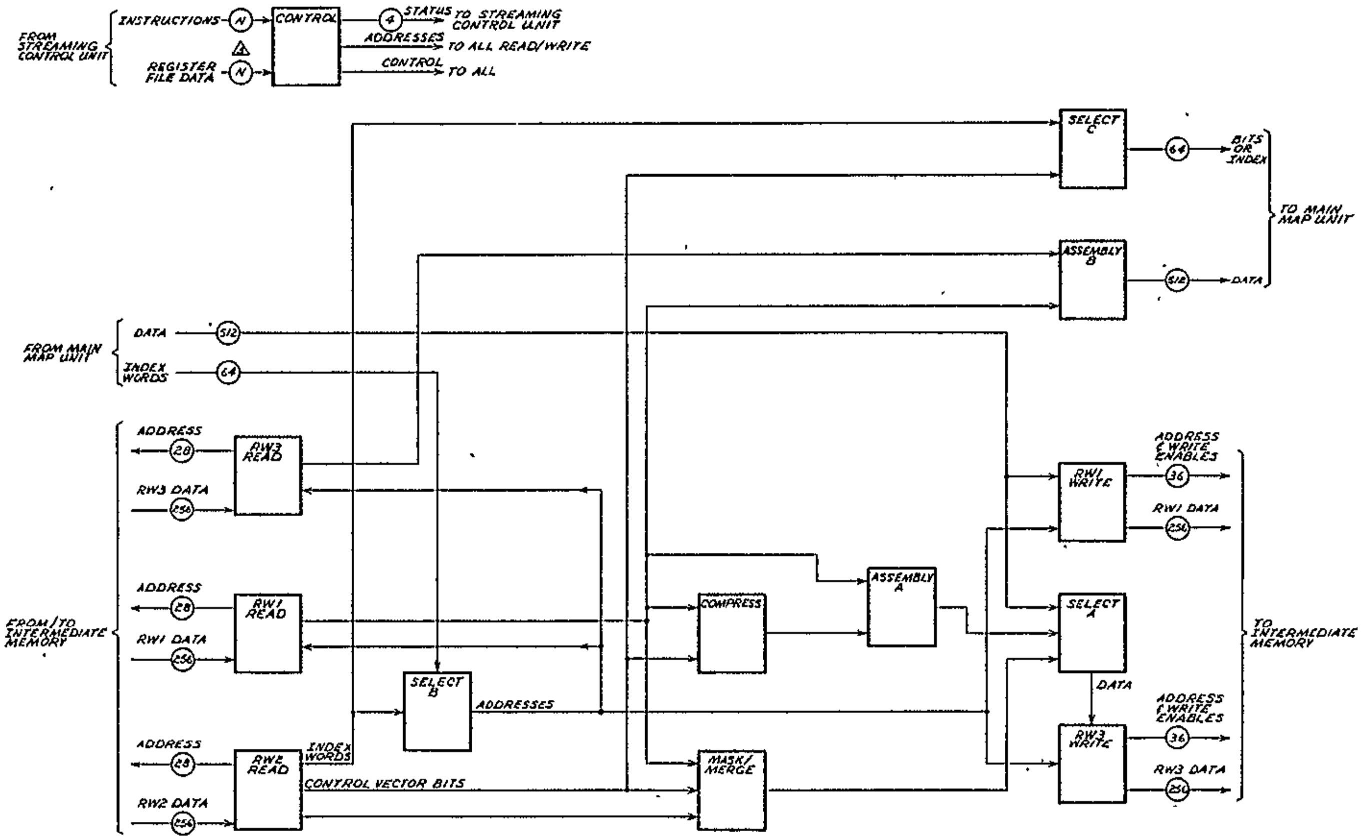


Figure 3. Main Map Unit



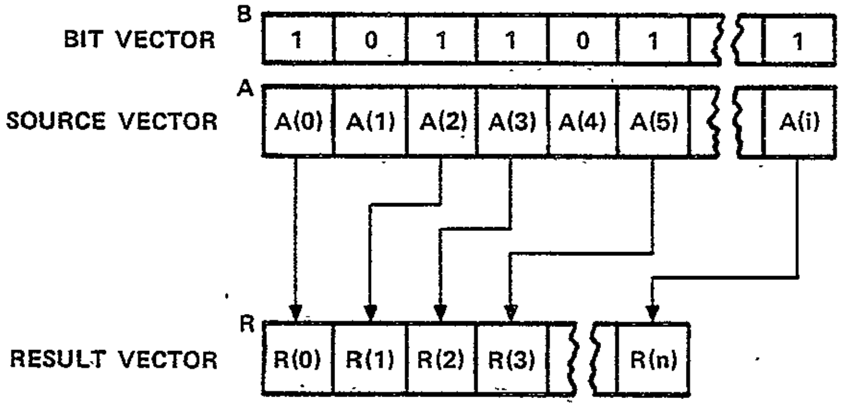
NOTES:

1. ALL DATA BUSES CARRY SECDED (NOT SHOWN IN BIT COUNTS).
2. RW"X" READ AND WRITE CANNOT BE USED SIMULTANEOUSLY

△3 UNDETERMINED NUMBER OF BITS

Figure 4. Intermediate Map Unit

1. COMPRESS--A linearly stored vector of input elements is input to the Map Unit, along with a binary string of bits, one bit for each 64/32-bit data element. Each bit of the bit string is examined in order and if it is a one, the corresponding data element from the input vector is transmitted to the result vector. If the bit is a zero the corresponding element from the input stream is discarded (not transmitted to the result vector). This is illustrated in figure 5.

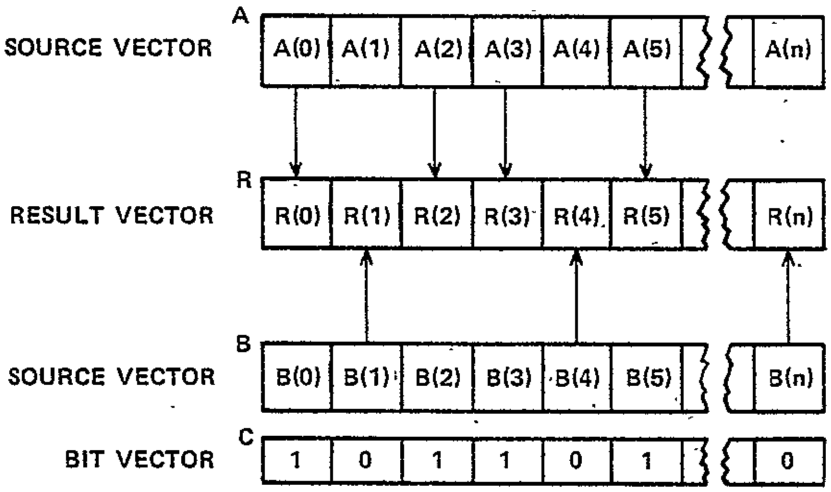


COMPRESS SOURCE WORD VECTOR A
BY BIT VECTOR B
GIVING RESULT WORD VECTOR R

COMPRESS ON "0"s IN B
(SELECT ON "1"s IN B)

Figure 5. Example of Compress

2. MASK--This operation inputs two vectors of data elements and a single bit stream. As shown in figure 6, the input data streams could be labeled A and B. The bit stream is examined one bit at a time. If the examined bit is a one the corresponding element from the A data stream is transmitted to the result vector, and the corresponding element from the B stream discarded. If the bit is a zero the corresponding element from the B stream is transmitted to the result vector, and the corresponding element of the A stream discarded.

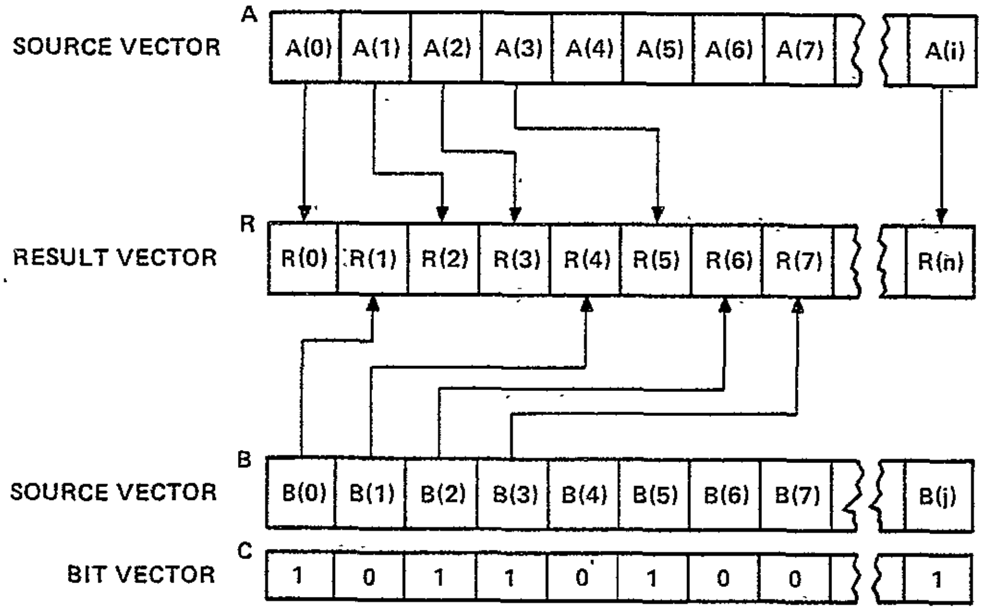


MASK WORD VECTOR A
AND WORD VECTOR B
UNDER CONTROL OF BIT VECTOR C
TO GIVE RESULT VECTOR R

SELECT A STREAM ON "1"s OF BIT VECTOR
(SELECT B STREAM ON "0"s OF BIT VECTOR)

Figure 6. Example of Mask

3. MERGE--This operation merges elements of two input data streams (A and B) according to a binary string of bits. (see figure 7) If the examined bit of the string is a one the next available element from the A stream is transmitted to the result vector; if the bit is a zero the next available element of the B stream is transmitted to the result vector. No element is discarded from either stream. The effect is to combine all elements of the input streams into a single vector.

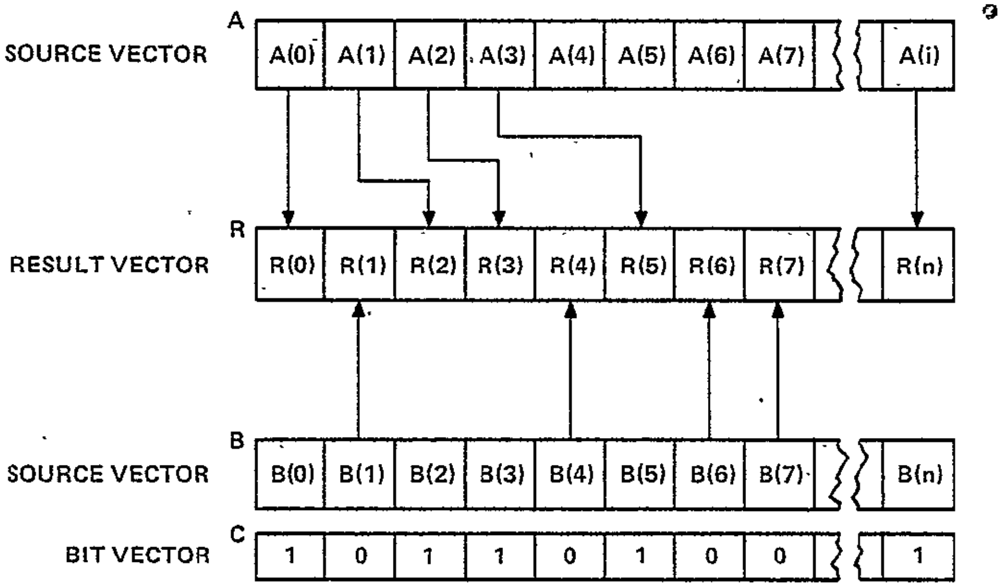


MERGE SOURCE WORD VECTOR A
AND SOURCE WORD VECTOR B
UNDER CONTROL OF BIT VECTOR C
TO GIVE RESULT WORD VECTOR R

SELECT A STREAM ON "1"s IN C
SELECT B STREAM ON "0"s IN C

Figure 7. Example of Merge

A variant of this instruction (shown in figure 8) discards B stream elements if the bit is a one, but does not discard A stream elements under any circumstances. The effect of the operation is to simultaneously decompress the A stream and insert the decompressed elements into the corresponding positions of the B stream.



DECOMPRESS IS A COMBINED
MERGE AND MASK FUNCTION

DECOMPRESS SOURCE WORD VECTOR A
AND SOURCE WORD VECTOR B
UNDER CONTROL OF BIT VECTOR C
TO GIVE RESULT WORD VECTOR R

SELECT A STREAM ON "1"s IN C
SELECT B STREAM ON "0"s IN C

A STREAM IS MERGED
B STREAM IS MASKED

Figure 8. Example of Decompress

These three operations facilitate the selection of data from a matrix according to either preset criteria (a prestored bit vector) or data/execution dependent criteria (there are several instrucions which generate the bit strings, called "control vectors" based on data comparisons by the Vector Unit). The compress, mask, and merge operations are not required for optimum performance of the 3-D implicit code, but are useful for the explicit codes and essential to weather and structures codes.

The key instruction in solving the implicit code on the FMP aside from the arithmetic operations are:

4. GATHER--The primary function of the Gather operation is to collect non-contiguous data elements into sequentially stored vectors which can then be processed efficiently by the Vector Units. The hardware can collect non-sequential records. A record is a group of sequentially stored elements which are accessed as a single entity by the Gather operation. Non-sequential single elements can also be collected by treating them as single-element records. Figure 9 illustrates the single-record Gather.

   The elements (or records) to be moved are selected either by means of a list of indexes, each of which points to a record in memory, or by means of a "stride", which determines the number of elements to be skipped before another record is selected. In figure 9 a fixed stride of 10 was utilized to cause the movement of elements 00, 10, 20, 30, 40, 50, 60, 70, 80, and 90 to the sequential vector X. The vector Y in figure 9 was formed by using a list of indexes which pointed to elements 340, 630, 570, 493, 294, 596, 699, 798, 897, and 697 of the original mesh. Note that the selected elements need not appear in any particular order and thus can be essentially random. In place of a single element the indexed list could point to records of data. For example the record length, RL, could be specified as 10 elements. The first record would begin at element 340 and would proceed sequentially through element 349. The next record would begin at the element pointed to by the next index, 630, and continue through element 639. Of course, in this case, result vector Y would be 10 times as long as in the single-element example.

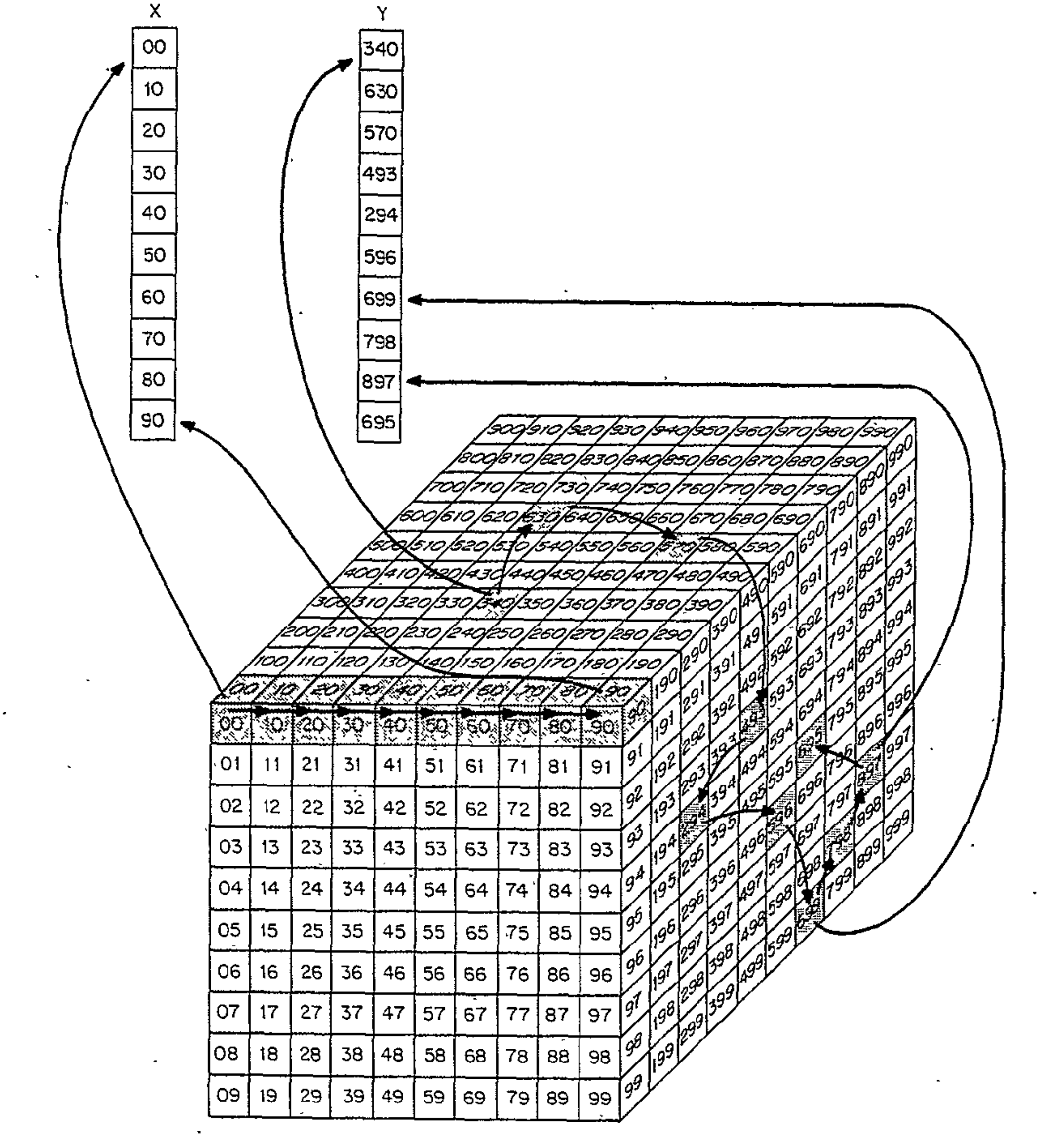


Figure 9. Examples of Gather with Fixed Stride = 10 and with Index List

In figure 10 the result vector is composed of records of length RL=10 taken from the mesh at stride intervals (ST=30) of 30 elements. The Map Unit then takes the first record beginning at element 00, moving ten elements for that record, then it applies the stride of 30 to the first element address and begins the next record at element 30.

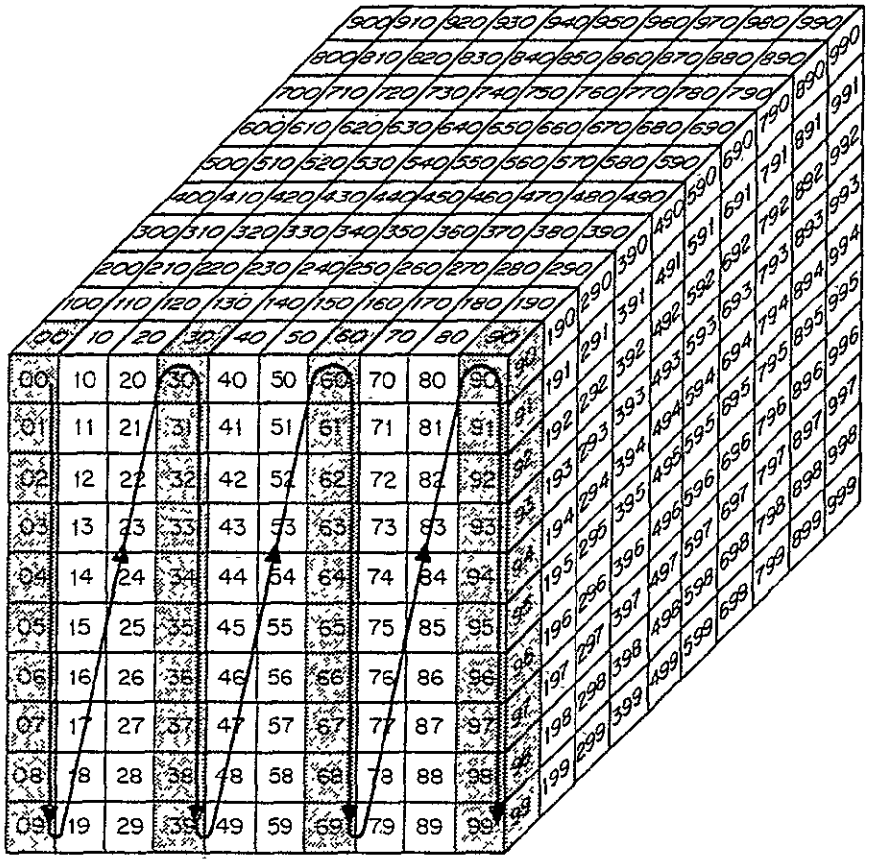


Figure 10. Gather Record with Fixed Stride = 30

The FMP provides for gather operations using strides in two directions (ST=n,m). Figure 11 shows the effect of using the two strides. First the hardware begins at element 00 and moves a record of length 1 (element). A stride of 10 is then applied (first stride) and the next record moved from element 10. Ten such records (NR=10) are moved, then the hardware restarts at the first element, applying the stride of 100 elements (second stride) and begins the process all over again until the specified vector length (VL=20) is filled. Strides may consist of any positive or negative integer value. It can be seen that with this instruction, two-dimensional and three-dimensional arrays can be transposed with a single gather instruction.

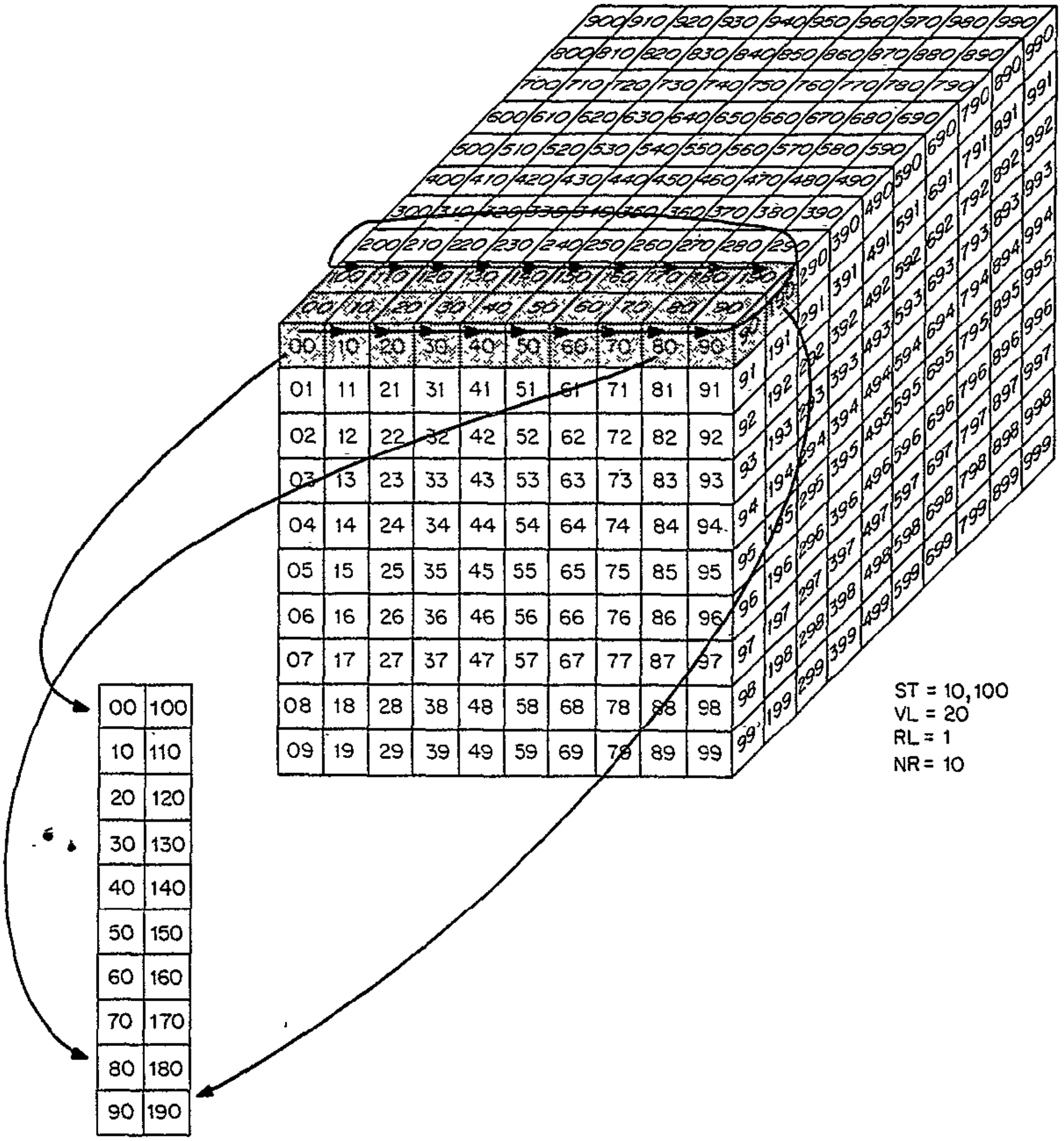
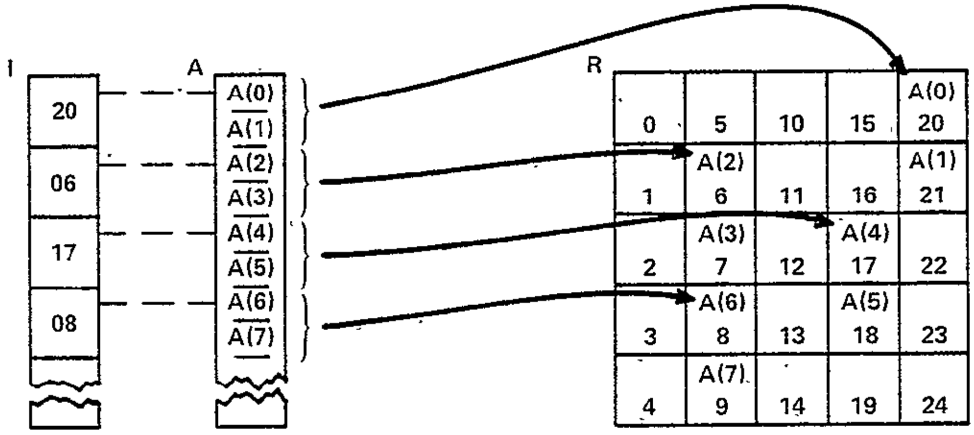


Figure 11. Gather Record with Two Strides

5. SCATTER--The scatter function operates as the inverse of the Gather in all of its options. Instead of collecting discontiguous elements however, this function distributes records (or elements) into memory according to the fixed stride or strides (ST=n,m) as specified in the instruction, or according to the list of indexes (for random storage of data). The scatter operation is illustrated in figure 12.



SCATTER WORD VECTOR A
TO WORD VECTOR R
USING INDEX VECTOR I

USE VECTOR A IN RECORDS OF TWO WORDS

EACH WORD IN I IS ADDED TO THE R
BASE ADDRESS TO COMPUTE A STORE
ADDRESS IN R FOR THE FIRST WORD
OF EACH RECORD

Figure 12. Example of Scatter

These five operations can be performed one at a time independently and concurrently in each of the Map Units, one operating with the Main Memory (LEVEL 1) and one operating with the Intermediate Memory (LEVEL 2). The two Map Units can be combined to perform Gather or Compress from LEVEL 2 to LEVEL 1 or Scatter from LEVEL 1 to LEVEL 2.

2.1.4 THE SCALAR UNIT

The control of the entire FMP ensemble is accomplished by centralized decoding of a single instruction stream in the Scalar Unit and distribution of the controls for Vector, Map, and I/O operations to other separate units. The Scalar Unit is described in detail in the functional specification found in Volume II. Since this element is the executing heart of the FMP, it has been decided that this unit should consist of the most mature design and technology available. The CYBER 203 Scalar Unit was chosen since it meets the requirements for performance and architecture ("distributed operation"), and possesses extensive diagnostic programs for a large machine structure.

2.1.5 THE VECTOR UNITS

The major arithmetic processing by the FMP is performed by the Vector Unit assembly (Vector Ensemble), which consists of four identical, separately controlled units (pipelines), plus one additional unit which acts as an on-line spare. One Vector Unit is diagrammed in block schematic form and discussed in detail in the functional specification (see Volume II). The key features of the Vector Units are:

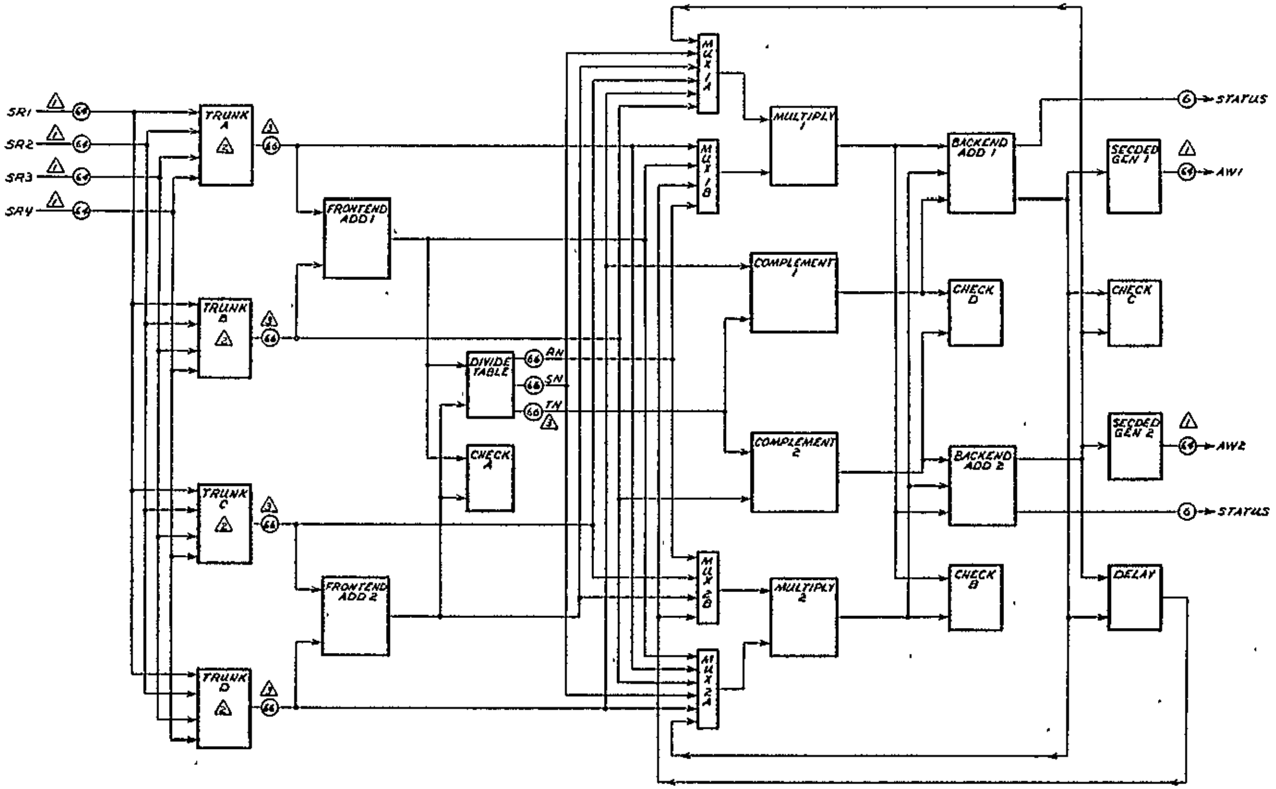
1) "Double Clocking"--The employment of pipelining of arithmetic operations makes it possible to use a faster clock cycle for these units than is required by the Scalar and Map Units. By using a clock period half that of the other FMP elements, the throughput of each unit can be doubled and the number of such units otherwise required reduced by half. This permits the designers to reduce the hardware components substantially over a normal "full clock" design. The reduction is not fully one-half, but rather closer to 35% since additional "latches" must be included in the design to hold operands between logic stages because of the speed of the faster clock.

2) "Variable Redundancy"--The reliability of the FMP is crucial to its success as a major facility. Since the Vector Units constitute a major portion of the non-memory hardware, the highest failure probabilities are then found in these units. Various techniques have been suggested (and discussed in more detail in refs. 1 and 2) for ensuring the validity of results

from the Vector Units. Two most prominent candidates were data parity (1 bit for every 8 bits of data) and modulus arithmetic. While these techniques provide proven validation of most of the data paths in an arithmetic unit they are not effective for the high speed multiply networks desired for the FMP, and they provide no surveillance of the variety of control networks engaged in the Vector Units.

A very attractive option is to provide total redundancy of Vector Units, each with its own control and data paths. Then a pair of units could check each other. This approach is fraught with two difficulties. First, the amount of additional hardware more than doubles the volume of parts (and hence interconnections) needed for the Vector Units, and thus increases the likelihood of component failure to unacceptable levels. Second, the impact of the additional hardware is seen in much higher machine cost, and additional chassis volume which can affect vector startup timing.

An engineering compromise is possible for this dilemma. The system could provide the completely redundant hardware which could be used for validation of answers, or for improving performance. The extent to which the redundancy is controlled would be a function of the programmer or compiler's intent or capability. The block diagram of the Vector Unit shown in figure 13 displays the use of redundant hardware as there are duplicate frontend adders, duplicate multiply units, duplicate complement networks, and duplicate backend adders in each Vector Unit. By providing four Vector Units of this type, the memory bandwidth can be matched and a minimum operation rate for 64-bit add, subtract and multiply can reach a useful limit of 500 megaflops with fully redundant checking of all arithmetic except for Divide. Since the Divide operation uses the same hardware (except for the divide table) as the Add/Sub operation, the redundant checking of the Add and Subtract is relied upon to verify the probable reliability of the Divide operation.



NOTES:

1. THESE BUSES CARRY SECDED (NOT SHOWN IN BIT WIDTHS).
2. INCLUDES SECDED CORRECT NETWORK.
3. DUPLICATE SIGN BITS.
4. ALL INPUTS/OUTPUTS ARE FROM/TO VECTOR STREAMING UNIT.

Figure 13. One Vector Unit

In figure 13 four checking units (CHECK A,B,C and D) are provided to compare the results from the various functional elements in the Vector Unit. Any combination of check units can be enabled for an operation, and if an error is detected by an enabled checker, a flag is sent to the Maintenance Control Unit (MCU) and the Vector Units are halted within six clock cycles of the failure detection. In the simplest case, where 500 megaflops is an adequate processing rate, data is fed from memory via buses SR1 and SR2 and selected through the corresponding trunks in the Vector Unit. This means that data on SR1 would be selected through TRUNK A and TRUNK C and data on SR2 would be selected on TRUNKS B and D. The identical functions and path selections would then be enabled for both halves of the unit. The vector operation

C=A+B

with A coming from SR1, B coming from SR2, and C going to memory on AW1, would follow the path through FRONTEND ADDER 1 and FRONTEND ADDER 2 with the corresponding results being compared in CHECK A. The unnormalized add results would then be passed through the multiply network and the corresponding results of this "pass" operation compared in CHECK B. The final postnormalization is performed in the backend adders and those results compared in CHECK C. The data then passes through the SECDED generator which appends 14 bits of error correction code (per 64 bits of data) and is transmitted to memory. Note that the complement networks are not engaged in this particular operation, however the data from trunks B and D (which should be identical) is automatically gated through these complement networks by the Vector Unit so that the results may be checked in CHECK D.

The above example demonstrates the use of total redundancy in the Vector Unit. To ensure that as much hardware as possible is being checked accurately, each element of the Vector Unit possesses its own control circuitry. The operation codes sent from the Scalar Unit carry a one-bit parity to each unit, which verifies its own operation code validity. All clocking, fan-out, fan-in, and microcode sequencing is then done entirely within that unit. This means that not only are the data paths verified, but the control sequencing and control fan-out are verified for each unit. This is a key part of the FMP Vector Unit design.

In many instances however, the 500 megaflop rate is not satisfactory, particularly since the FMP objective is a sustained rate of at least 1000 megaflops. To achieve higher processing rates some of the additional hardware in each Vector Unit must be brought into play. Take as an example

R=(A+B)*C

with A coming from SR1, B coming from SR2, C coming from SR3, and R returning to memory on AW1.

The A and B operands are processed by FRONTEND ADDER 1 while the C operand passes through FRONTEND ADDER 2. Since the results coming from the two frontend adders cannot be identical in this case, CHECK A is turned off. The sum A+B is then fed to MULTIPLY 1 via MUX 1B and the multiplier C is fed through MUX 1A. At the same time the frontend adder output is fed to MULTIPLY 2 via MUX 2A, and the multiplier is transmitted through MUX 2B. Thus the multiplier results can also be checked in CHECK B. By selecting SR2 through TRUNK D as well as TRUNK B, identical data will be fed through the complement network (although it has no use in this particular operation) and thus the results can be compared in CHECK D. The results of the multiply operation are post-normalized in BACKEND ADDER 1 and also BACKEND ADDER 2 so that they may be compared in CHECK C. The resulting normalized results are then sent to memory after SECDED has been generated.

In this example, three pairs of the four sets of arithmetic elements are checked for validity at each clock cycle. Depending on the operation desired and the processing rate required, the amount of checking can be varied from 100% to no worse than 25% of the actual hardware in the unit. Since an operation such as

R=(A*B)+(C*D)

will have different results emerging from the corresponding multiply elements, frontend adders, and complement networks, it might be desirable in some critical cases to force the object code to break up the operation into three parts, at a consequent loss in performance, in order to assure the validity of the answers:

```
T1=A*B
T2=C*D
R=T1+T2
```

This technique can be quite costly in storage space allocation however, and given a large mix of dyadic and triadic operations in the Vector Units, it can be expected that on a probabilistic basis the confidence in the results should be close to 100%, if the variable redundancy technique is used. It is obvious that the FMP compiler should provide a compile time option which restricts the generation of object code to simple monadic operations where a programmer wishes to achieve 100% checking of results.

The additional spare Vector Unit is physically connected to the data trunks at all times. The trunk network can be electronically switched by the Maintenance Control Unit (MCU) to extricate any failing Vector Unit, and reconfigure the system so that four non-failing units (including the spare) are placed back on-line. Since the spare unit is always connected to the trunk, it can be made to behave in identical fashion (function and data) with one of the other units, except for returning results to memory.

This unit is then checked in a continuous manner as are the other operating units. A comparator is also provided which compares the outputs of the spare unit and the unit which it is tracking; the MCU is notified of any non-compare. In addition, the spare unit can be logically isolated from the other units and then driven by the MCU (at a greatly reduced rate), returning results to the MCU, such that it can be at least partially diagnosed while the other units are performing useful work.

The Vector Units are capable of operating in 64-bit, 32-bit or mixed mode. When operating in 32-bit mode all arithmetic functions except divide can produce two floating-point results per cycle. Thus, a single Vector Unit can perform

```
A*(B+C*D)
```

in 32-bit mode and effectively yield 3 operations * 2 operands = 6 results per cycle on one of two output ports. (The other output port can provide a partial result of that which appears at the first port, dependent on what the operations are; this is not considered in this example.) Using existing, high performance LSI technology, this cycle for the Vector Units can realistically be set at 8 nanoseconds. Four Vector Units would then produce 24 results every 8 nanoseconds, or 3 results every nanosecond, for a peak operations rate of 3000 megaflops.

2.2 THE FMP OPERATION

To illustrate the operation of the FMP a sequence of code is extracted from the three-dimensional implicit solution (appendix B):

```
930    RJ=Q(2:KMAX-1,L:L+LSM,6,*)

940    XKL=X(*,L-1:L+LSM+1,2:JMAX-1)

950    YKL=Y(*,L-1:L+LSM+1,2:JMAX-1)

960    ZKL=Z(*,L-1:L+LSM+1,2:JMAX-1)

970    XK=(XKL(3:KMAX,2:LSL+1,*)-XKL(1:KMAX-2,2:LSL+1,*))*DY2

980    YK=(YKL(3:KMAX,2:LSL+1,*)-YKL(1:KMAX-2,2:LSL+1,*))*DY2

990    ZK=(ZKL(3:KMAX,2:LSL+1,*)-ZKL(1:KMAX-2,2:LSL+1,*))*DY2

1000   XL=(XKL(*,3:LSL,*)-XKL(*,1:LSL-2,*))*DZ2

1010   YL=(YKL(*,3:LSL,*)-YKL(*,1:LSL-2,*))*DZ2

1020   ZL=(ZKL(*,3:LSL,*)-ZKL(*,1:LSL-2,*))*DZ2

1030   XX(1)=(YK*ZL-ZK*YL)*RJ(*,*,2:JMAX-1)
```

Found in lines 930 through 960 is a sequence of map operations which move data from Intermediate Memory (LEVEL 2) to Main Memory (LEVEL 1) as gather record operations. The stream of instructions is delivered to the Scalar Unit which first performs whatever scalar setup of addresses and lengths is necessary for the first map function (line 930). In this case it consists of computing the length of the records to be transmited since the array Q is dynamic and its dimensions must be computed at object time. The base addresses for the map operations must also be computed at the same time. The map operations is then sent to the Streaming Control Unit (SCU), which is assumed to be idle at the moment.

Since the map operation transmits data between the LEVEL 1 and LEVEL 2 memories, the Streaming Control Unit engages both Map Units, set up the data paths, and transmits the appropriate internal functions to both units. While this process is underway the Scalar Unit continues executing scalar instructions which perform the interpretation of statement 940, computing record lengths and base addresses and transmitting the map instruction to the Streaming Control Unit. Since the Map Units are now busy moving data for the fist operation, the incoming map instruction is queued in the Streaming Control Unit. The Scalar Unit continues instruction decode after this second map operation is accepted by the SCU. A third map operation is then set up and transmitted to the Streaming Control Unit. A fourth map operation is set up, transmitted, and queued by the SCU. The Scalar Unit continues processing the instruction stream, which probably contains other scalar setup operations, until the vector arithmetic called out by statement 970 is encountered. This vector instruction is set up and sent to the Streaming Control Unit. However, this instruction contains a dependency key of "01" which prohibits execution until the corresponding key becomes not busy. Since this key was assigned as a write key to the second map instruction, the vector instruction will wait in the SCU until this key becomes not busy, signifying that the corresponding data (Vector XKL) has been completely mapped into Main Memory.

The Scalar Unit proceeds to execute more instructions until the next vector operation (statement 980) is encountered. Since the first vector operation is held up and not yet in the streaming queue, the Scalar Unit cannot issue any more stream instructions to the SCU. The Scalar Unit then pauses until the previosly issued vector instruction becomes free of its dependency key conflict. Upon completion of the map operation for Vector XKL, the next map operation is immediately commenced (for Vector YKL). The vector operation is then initiated on the XKL data, the Scalar Unit then can issue the vector instruction for statement 980. Since the map operations proceed at a much slower rate than the vector operations, the vector operation at 970 will be done before the map operation which moves YKL data to Main Memory is complete. The next vector operation is then held up for its data (Vector YKL) by the same dependency key mechanism (although with a different key) as described previously. In this instance again, the Scalar Unit will pause until the dependency key becomes not busy. This same process continues for the Vector ZKL.

The sequence of pauses illustrated herein obviously affect the performance potential of the FMP. For that reason the actual code generated for this example includes prefetching the vectors XKL, YKL, and ZKL with the Map Units long before they are needed by the vector arithmetic. In actual fact, the fetching of the next set of data for these arrays is carried on concurrently with the vector arithmetic on the current XKL, YKL, and ZKL data.

Once the vector operation at statement at 990 is initiated by the Vector Unit, because its dependency key has become non-busy, the remaining vector operations are set up and queued in the Streaming Control Unit without conflict since no dependency key is needed (the data is already known to be available).

Statements 970 through 1020 illustrate the use of two functional elements in each pipeline (a subtract operation followed by a multiply), which yields a floating-point rate of 1000 megaflops peak value. Statement 1030 performs two multiplies and one subtract in one instruction and thus yields 1500 megaflops peak rate computation in 64-bit mode. The remaining multiply by RJ is combined into a later vector arithmetic operation.

2.3 RATIONALE SUMMARY

The FMP hardware described in this report represents an example of a processor which, in the best judgment of RADL personnel, can be built and utilized in the period beginning in 1983-1984, and which will meet the performance and reliability goals (1 gigaflop, 95% availability) established as minimum by NASA-Ames. The final form represented here is necessarily the result of many complex tradeoffs involving schedule, timing, technology, manufacturing considerations, and the crucial reliability, availability, and maintainability (RAM) factors that must be taken into account for such a large assemblage of equipment. Some of the architectural decisions are obviously linked to Control Data's experience with the STAR-100 and the CYBER 200 family of Vector processors. A further, practical linkage involves the use of systems and diagnostic software for the FMP that can be derived at minimal risk from the CYBER 200 systems. The thought process that has been involved in these design decisions has extended through prior study periods into this present study with the appropriate rationale documented in previous reports (refs. 1, 2). For the most part these rationale remain unchanged from the previous studies but, since they are so critical to the outcome of the project, it is desirable to restate them here in order of priority to FMP success.

2.3.1 RELIABILITY, AVAILABILITY, and MAINTAINABILITY

Although the FMP was founded on the premise of producing the fastest computer in existence in the 1980's in exchange for some "tailoring" of the hardware to match specific algorithms, the major consideration throughout the FMP design project has been that of reliability. The corollary issues of availability and maintainability were also included in this area of priority concerns. In another section of this report these topics are defined and discussed in more detail. Here the effect of these considerations on the FMP structure will be covered.

The overriding concern of designers for a system as large as the FMP is the parts count and the effect of this count on RAM. The hierarchial memory was arrived at (as previously discussed) by evaluating the performance and capacity requirements arising from the characteristics of the flow models. The high performance Main Memory was specified on the basis of the most powerful memory system currently available, the CYBER 200 two million word, 128 billion bits per second, central memory. This memory system utilizing the recently available 4096-bit RAM chip, can hold 8 million words and achieve the same high performance levels as the 2 million word system. This memory requires over 175,000 parts. If a larger memory, say of 32 million words, were desired then over 700,000 parts would be required. Given a nominal failure rate goal of .01% of all parts per 1000 hours that would mean 70 failures per 1000 hours or a failure evey 14 hours. Even with single-error correction, double-error detection (SECDED) networks protecting the memory, the FMP would encounter an unacceptable level of interruptions as the probability of a double-bit error occurring becomes quite high.

At the expense of lower performance, another level of memory could be built of 65K-bit chips to produce a memory system of 32 million words with only 38,000 parts (approximately).

Finally, if 128-256 million words are necessary for the solution of some research problems, the 65K part again becomes inadequate to the task as so many parts are needed in the memory that the failure rate reached intolerable levels, despite the employment of SECDED. Another storage hierarchy is thus indicated using higher capacity, lower performance memory components. If the predicted 256K-bit CCD memory part becomes available, the memory part count becomes approximately 36,000, with a concomittantly acceptable failure rate. The realistic production of 1 million bit bubble chips (which would reduce parts count for this level of memory to around 9000) appears to be possible in the 1982 timeframe. The design goals for LEVEL 3 bandwidth currently preclude the use of bubbles however, because of the relatively slow serial transfer rate of bubble technology.

2.3.1.1 EFFECT OF PARTS COUNT

The table below gives approximate chip counts for the major elements of the FMP. Chip types are as follows:

- CPU - LSI and Other (microcode memory/high speed buffers)
- Main Memory - 4K x 1-bit bipolar RAM
- Intermediate Memory - 128K x 1-bit MOS RAM
- Backing Store - 256K x 1-bit CCD

The memory chip counts include approximately 10% which are for control and interface; the balance is the indicated type of memory chip.

Estimated Chip Count for FMP

Element	Chip Count
CPU	11K LSI* 13K Other*
Main Memory	185K
Intermediate Memory	20K
Backing Store	40K

* LSI is the CDC LSI-168 gate array; other chips are for microcode memory and high speed buffers.

Overall reliability is determined largely by parts count and that effect influences design decisions about the memory. What are the other effects of parts count? First, the FMP logic should be examined to see how reliability can be affected by parts count in the remainder of the CPU. Control Data chose for the FMP the most technologically aggressive circuit family, in terms of density and speed, that can be expected to be available in the period identified for construction of the NASF. This family consists of Large Scale Integration (LSI) circuits of speed on the order of 700-900 picoseconds for typical logic elements. With this component the Scalar, Vector, Map, Memory Interface, and Streaming Control Units can be built using about 18,600 parts. This design would involve the building of nine Vector Units, each operating at a 16-nanosecond clock cycle. Although 18,600 parts are a relatively small number compared to the Main, Intermediate, and Backing Store Memories, it must be remembered that most of the memory parts are protected by SECDED, while a good deal of the CPU logic is not. By double-clocking the Vector Units (described previously) it is possible to construct the FMP with only five Vector Units (four operating units and one spare) with a consequent reduction in LSI parts of approximately 5400 or about 25%.

A second effect of parts count is the need for interconnection of the parts involving solder and pressure connections, bonds, and metalized paths. The impact of interconnections is secondary to part failure rate in the reliability calculations, but still significant for a large ensemble such as the FMP (refer to Division 6).

A third effect of parts count is subtly linked to the availability and maintainability of a hardware network. Once a failure occurs, what is the probability that it can be corrected automatically, thus requiring no emergency maintenance activity? Further, if a maintenance action is required, what is the probability that the failing part can be isolated within a minimal time objective "TR" (Time to Repair)? These probabilities directly influence the probable RAM (reliability, availability, and maintainability) objectives for the FMP.

By basing the FMP on a large, homogeneous memory system, the designers have attempted to make maximum use of memory characteristics to affect the RAM. Memory, delightfully, consists of well ordered parts with limited interconnection. The use of SECDED on memory data not only provides a first level of defense against memory failures (by automatically correcting single errors), but provides information (via the check bits) which can be analyzed by the Maintenance Control Unit (MCU) to assist Service Engineers in the isolation of the failing components in minimum time. While there are failure modes in the memories that cannot be detected or corrected by SECDED (such as power bus, address line, and read/write strobe failures) over 98% of the memory parts are covered by SECDED.

SECDED should be carried throughout all data paths wherever possible to provide automatic correction, as well as detection, to the maximum extent. In the FMP, SECDED has been carried into the Vector Units up to the point where the data enters the unit and the check bits can no longer be retained (the data will be altered by arithmetic operations). SECDED is regenerated for results emerging from the Vector Units being transmitted back to memory. Double failures in these data paths (in the Map and Vector Units) will yield check bits that can aid the engineer in fault isolation in minimum time.

2.3.1.2 TRANSIENT ERRORS

A word must be said about the most infuriating culprit in large systems, transient errors in data and control. A totally failing component will generally make itself known rather quickly, either through the mechanism of the SECDED error detection system, on-line diagnostics, or abysmal failure of a "stable" production code. In large complexes of hardware such as the FMP, the possibility for transients occurring due to induced noise, bus fluctuations, marginal part operation, vibration, and perhaps such magical influences as gamma rays must be considered despite the best engineering efforts at shielding, margin testing and power system overdesign. Since a good portion of the system will be protected by SECDED, the effects of transients in the system will be invisible, except for a random, uncorrelated error report made by the SECDED networks to the MCU. There are times however, when a "hard" error in a data path, due to a component failure, will be correctable by SECDED except when a coincident transient appears. The probability of this situation occurring is dependent on the probability that at any one time there will be failing, but correctable, component errors in the system. This situation further depends on real failure rates and maintenance strategies.

Remembering that a double SECDED error will cause a system interruption, one must evaluate the frequency of maintenance replacement of components being compensated for by SECDED and the possible existence of transient errors occurring which could cause double errors to appear. A beginning maintenance strategy is therefore projected which minimizes the number of failed components being left in the machine in a given 24 hour period. As experiential data is accumulated, it may well be possible that the probabilities of transient errors, or coincident double component failures, in a given network may permit a more liberal maintenance policy, with consequent cost savings. At present, with the rate of transient failures impossible to determine until the hardware system is built, it seems necessary to specify the most conservative maintenance strategy with its attendant high costs (see maintenance study report and maintenance action assumptions in Divisions 6 and 7).

Another, and potentially more dangerous, consequence is the possibility of undetected transient errors occurring which affect critical result data. FMP users must be able to depend on their solutions without having to run a problem three times to set a majority vote on the most probable correct answer. As stated previously, a solid component failure should make itself evident during on-line diagnosis of the FMP, which is performed periodically during job execution. A solid, uncorrected, component failure might occur during a particular solution execution causing some or all results to be invalid. Generally, before these results can be propogated to other jobs or users, the on-line maintenance diagnostics would have found the error and warned the installation that jobs run since the last diagnostic pass are probably specious. While it is acceded that a totally redundant system would make the user instantly aware of the possibly invalid results, the cost and parts count for such a system make it prohibitive to build.

A transient error causes more havoc, however, since it may occur at a time when diagnostics are not being run, or at a place that cannot be checked by SECDED or by the Vector Unit's variably redundant comparators. Results from such runs would contain invalid data with no warning as to the fact that an error occurred which negates the particular run.

There are then two kinds of undetected errors that can yield bad results. One which can be diagnosed at a later time, and hence cause the user to invalidate that set of answers, requires some degree of systems management and human action in evaluating the diagnostic messages to determine what, if any, results should be discarded. The second kind of undetected error will not be known to the user, but it will be based on the very small probability that a transient error occurs under undetectable circumstances. With over 95% of the hardware networks in the FMP being protected by SECDED and by the checking of the Vector Units, and with on-line diagnostics being employed on a regular basis to ensure a certain level of confidence is maintained in the machine, the probability of producing undetected errors in results is necessarily unknown but theoretically should be extremely low.

The effect that these types of failures (undetected by SECDED or vector checkers) have is to require a portion of the FMP power to be expended on a continuous basis throughout the operating day to establish a minimum confidence level. The interval and extent of diagnostic execution is determined by the maximum allowable period of time before results must be flagged invalid, and the time required for diagnostic operation to achieve a threshold of confidence. Thus, diagnostics become an additional "job load" on the FMP that must be taken into account when evaluating the total system throughput. (See discussion in System Simulation section later in this volume.)

2.3.2 BUILDABILITY

The second most important consideration in creating the FMP is ensuring that the machine can be built at all in the timeframe required. A major factor, of course, is the parts and interconnect count described previosly. Obviously, it would be possible to conceive a machine that would entail the assembly of so many components that the sheer volume of soldering, bolting, and hookup exceeds the limit of errorless construction. The resulting chaos involved in removing fabrication errors (differentiated from component failures) might prove to overwhelm the manufacturing operation.

A second and equally important factor in determining buildability is the choice of component technology, not only electronic but mechanical, power, and cooling. The choice of circuit components naturally determines the rquirements for cooling and power, while influencing packaging decisions meant to maximize the density of circuitry for performance and space reasons.

The original feasibility of the Control Data FMP was based on a postulated family of LSI circuits with densities of 500 gates and speeds under 500 picoseconds per gate. The painful process of creating a high technology, such as high speed LSI, and carrying it through volume production, made it clear by the second phase of this study that a 1982-1983 built FMP would have to be constructed out of extant technologies. Thus effort was applied to increase the parallelism of the architecture to achieve the gigaflop goal with existing circuit and packaging families. The culmination of this decision is found in the design described in this report. Certainly, if a family of logic could be found that was twice as fast, then the number of parallel units could be halved with the very desirable reduction in parts. Certainly, if a family of memory components could be found with densities of 2 to 4 times that of the existing technologies the parts counts, failure rates, and probably cost of the memory systems could be reduced accordingly.

It is the inevitable hope of designers and consumers of the FMP that the system could benefit from the most up-to-date, aggressive technologies available. To that end, each of the studies has examined technology futures with an eye to employing new developments in the FMP. Unfortunately, the choice of technologies cannot be delayed until after all design is complete and the machine is about to be constructed. The technological choices are integral to the initial architectural approach, the development of tools to support design and construction, design techniques, and the detailed design itself. Therefore, the FMP described in this report is heavily predicated on the use of the family of parts, 4K bipolar RAM, 65K MOS RAM, 256K CCD memory, and Fairchild F200K logic. With this family, a machine can be built to meet the minimum reliability and performance goals established by NASA. Any major change (doubling speed or density, for example) in available technology would make a reassessment of the existing design from the ground up an essential task in an effort to reduce cost and parts counts.

The technolgical possibilities that appear to loom over the 1980 horizon are tantalizing to consider for the FMP if, and only if, the FMP were to be constructed, say, beginning in 1984-1985. Not only would solid progress be evident on high speed silicon parts, but the potential of the gallium arsenide technology should be proven (see technology update report, Division 5). It would then be conceivable that a machine with a two gigaflop computation rate and half the logic hardware might be built at very reasonable cost. The difficulty with proposing a delay in FMP development to await these "futures" to come to fruition is that there are too many unpredictable factors that can effect the outcome of such strategy:

a) There is no doubt that the scientific problems presently known in development of either higher density/speed LSI or gallium arsenide logic components can be overcome. The essential question is -- in what time frame can they be solved?

b) The willingness of vendors such as Motorola, Fairchild or Texas Instruments to make the resource commitments and capital investments necessary to bring new, high technology devices into production is based only partly on technical feasibility. The projected profitability of a particular manufacturing line (based on volume and price expectations), the capital outlay requirements, and the general state of the economy are governing factors on the availability of the high performance, high risk, essentially low volume components that designers seek for FMP-type machines.

The effect of these issues on the architecture and design is obvious. A second, less visible, effect is the development time and resources needed to create and productize the tools necessary to utilize the new technologies. The time has passed when an engineer could gather a box of transistors (or small scale integration chips), sketch out a piece of design, and breadboard the affair with his own soldering iron. The very nature of LSI means that designers commit whole ensembles to single silicon slices requiring complex steps in manufacture. Design analysis and simulation software for a particular technology must be in place and operational before that technology can be effectively employed. There is a definite lead-time then between technology selection, useful design and simulation, and final circuit components that can be as long as five to seven years. The significance of these lead-times and uncertainties is that although there may be some technological "magic" awaiting the computer community in 1984-1985, it is not possible to consider using such components for the NASA FMP without incurring grave risk to schedule and buildability. Hence, the somewhat conservative choice of

Fairchild LSI logic, for which a vast assemblage of design and manufacturing software and procedures is now available.

2.3.3 PERFORMANCE

Not last, or even least, in consideration is the element of performance for which the FMP project was originally created. At the outset, a minimum threshold of one billion floating- point operations per second was established for solutions of the Navier-Stokes equations. In theory, this level could be achieved by a single processor operating at a clock cycle of one nanosecond, or 1000 processors operating at a clock cycle of one microsecond. In practice, a one-nanosecond cycle time for floating-point operations on numbers with 48-bit coefficients is not yet achievable, while the harnessing of 1000 slow processors creates massive headaches in design of interconnection and control, not to mention programming. These issues have been discussed at length in preceding reports. The major tradeoffs in memory systems capability, number, and speed of vector units and performance of circuit technology have also been covered in previous reports as well as preceding discussion in this report. In the aggregate then; the search for performance has involved:

a) determining the peak vector arithmetic rate to support a <u>sustained</u> solution rate of at least 1 gigaflop;

b) designing the minimum hardware conglomeration to provide the peak vector rate, and minimize vector startup;

c) isolating those functions in the implicit and explicit code which limit the FMP from maintaining the sustained rate;

d) designing a fully concurrent map unit system to perform those non-arithmetic tasks in parallel with computation;

e) designing a memory system that could supply the peak vector bandwidth plus the data rates needed by the map units;

f) testing the resulting structure with code sequences taken from the various flow metrics;

g) reworking the design to improve the performance of those limiting cases that are of significance;

h) testing the programmability of the flow codes with the resulting structure;

i) going back to step c) and trying again.

The conclusions reached for each of these items form the basic rationale for the design of the FMP as it now appears:

a) At the outset it was assumed that the arithmetic processing bandwidth of the FMP would have to be substantially greater than the minimum threshold of 1000 megaflops in order to arrive at a sustained rate of that minimum. This premise arises from the experience with existing high performance computers in actual use. The theoretical peak rate of the Control Data 7600 is limited to the issue rate of a new instruction every 27.5 nanoseconds which yields at best 36 megaflops. The measured peak rate for memory-to-memory operations, however, is closer to 12-15 megaflops, due to the loads, stores, and inter-register transfers. The nominal rate assigned to the 7600 for production codes is 3-5 megaflops in actual use. The STAR-100 possesses a peak vector rate of 50 megaflops in 64-bit mode. In certain large production codes, the sustained rate is shown to be 20 megaflops which is a reduction of 2.5 to one over the peak rate. Other competitive machines demonstrate a similar degradation of 2-3 times from peak performance for a sustained rate.

   These observations led the FMP architects to set vector processing rate goals of 2-3 gigaflops peak rate for initial design evaluation. If later it proved possible to improve the ratio between sustained and peak rates, the 2-3 gigaflop objective might be reduced somewhat.

b) The hardware necessary to meet the 2-3 gigaflop peak rate could have been chosen from a variety of options, but pipelining was selected for two practical reasons:

   1) Parts and interconnection count analysis indicated that pipelines could be built with fewer of these critical items than a multiplicity of processing units yielding the same compute power.

   2) A substantial amount of design and development had already been completed on suitable pipeline elements for the Control Data CYBER 200 family of supercomputers.

An assemblage of 32 identical arithmetic pipelines, each operating at 16 nanoseconds to achieve a 2-gigaflop rate is a perspicuously brute-force approach to the problem. The volume of hardware this would entail seriously strained the limits of buildability, which had been given higher priority than performance. The limit of pain in hardware seemed to indicate that eight pipelines, which would have the ability to perform more than one arithmetic process per clock cycle demanded the maximum allowable componentry. The need for error checking led to the "variably" redundant structure that has been described previously, while the desire to have still fewer circuits motivated the design of "doubly" clocked units. As a result of these deliberations, the final Vector Units could deliver 64-bit result operands to memory at the rate of one per pipeline for each of four pipelines every eight nanoseconds.

Since this represents a result rate to memory of only 500 megaflops, how can the units be alleged to maintain rates commensurate with the 1-gigaflop requirement? Analysis of the three-dimensional implicit code showed that a majority of arithmetic expressions involved an average of three processes:

From the BTRI subroutine (see appendix B) line 4450:

```
U25=(B1(2,5)-L21*U15)*L22
```

If each pipeline could perform all three operations on the data for each result (U25) returned to memory per clock cycle, then the arithmetic result rate could be bumped to 3*500=1500 megaflops. Using this technique as a basis, the implicit code was analyzed to determine if this "triadic" facility would be fully utilized. It was found by hand calculated estimates that the FMP would probably operate at an average rate of 1200 megaflops for the whole code, assuming the data could be delivered to and removed from the pipeline to match that rate.

In some instances it is not possible to keep the maximum number of arithmetic elements in the pipeline busy with "triadic" or "dyadic" operations. For example, from lines 1070 and 1080 of the implicit code (see appendix B)

```
1070  D(1,2)=XX(1)*HDX
1080  D(1,3)=XX(2)*HDX
```

two separate vector operations would normally be generated by the FORTRAN compiler, with a consequent result rate of 500 megaflops for each separate multiply. The FMP provides two separate data trunks to memory and four separate data trunks from memory which permits more intelligently generating one instruction to perform both multiplies concurrently. In this case the four pipelines behave as if they were eight, and no data comparison is done by the CHECK networks in the pipelines.

Another use of the two separate result streams to memory is the storing of partial results on one trunk and full results on another. For example, take the sequence from the BTRI subroutine, lines 4720 and 4730:

```
4720  D1=L11*C1(1,M)
4730  D2=L22*(C1(2,M)-L21*D1)
```

wherein the result D1 could be stored to memory on the AW1 result trunk while the partial result (C1(2,M)-L21*D1) could be stored via AW2, for later use. Note that in this case the capability of the Vector Units for performing three arithmetic operations per clock cycle is achieved by a simple compiler technique of combining functions from "common subexpression analysis".

The 1.2 gigaflop rate is, of course, not the 2-3 times safety factor the designers had been seeking. Achieving that would require doubling the hardware complex to 8 of the triadic, eight-nanosecond pipelines to yield 2*1.2=2.4 gigaflops. This would in turn require a doubling of the memory bandwidth at the cost of additional hardware and trunks that again strain the buildability constraints. Two other avenues were open to ameliorate this situation:

1) Use of 32-bit arithmetic--The CYBER 200 family provides a dual arithmetic system which is used for improving throughput, as well as data storage, for those problems that can tolerate the reduced significance. The design of pipelines for this dual mode demonstrates that the additional hardware needed to split a 64-bit pipeline into two 32-bit pipelines, dynamically and when the occasion warrants, is insignificant. In the CYBER 200 structure such pipelines can then deliver two 32-bit results for every one 64-bit result, in most cases doubling the result rate of the pipelines. If this technique is applied to the FMP, then four FMP pipelines could produce a peak rate of 8 result operands delivered to memory every 8 nanoseconds. This means that if a single arithmetic operation is performed for each result, a minimum of one gigaflop is reached in 32-bit mode. However, when the triadic capability of the pipelines is applied to 32-bit mode, a peak rate of 3000 megaflops is attainable. In the projected case for the implicit code this would mean a result rate of 2*1.2=2.4 gigaflops computation rate. At least for 32-bit cases this architecture, with minimal hardware, provides the purported 3 gigaflop peak rate. The determination of whether or not certain computations can be done in 32-bit mode seems to elude even the most expert numerical analysts, and most probably must await empirical testing of algorithms in both modes. Since in all likelihood computations could most effectively be done in a mixture of 32-bit and 64-bit forms, depending on knowledge of the numerical behavior of a given calculation, the FMP Vector Units have included the ability to perform operations on mixed 32/64-bit data streams.

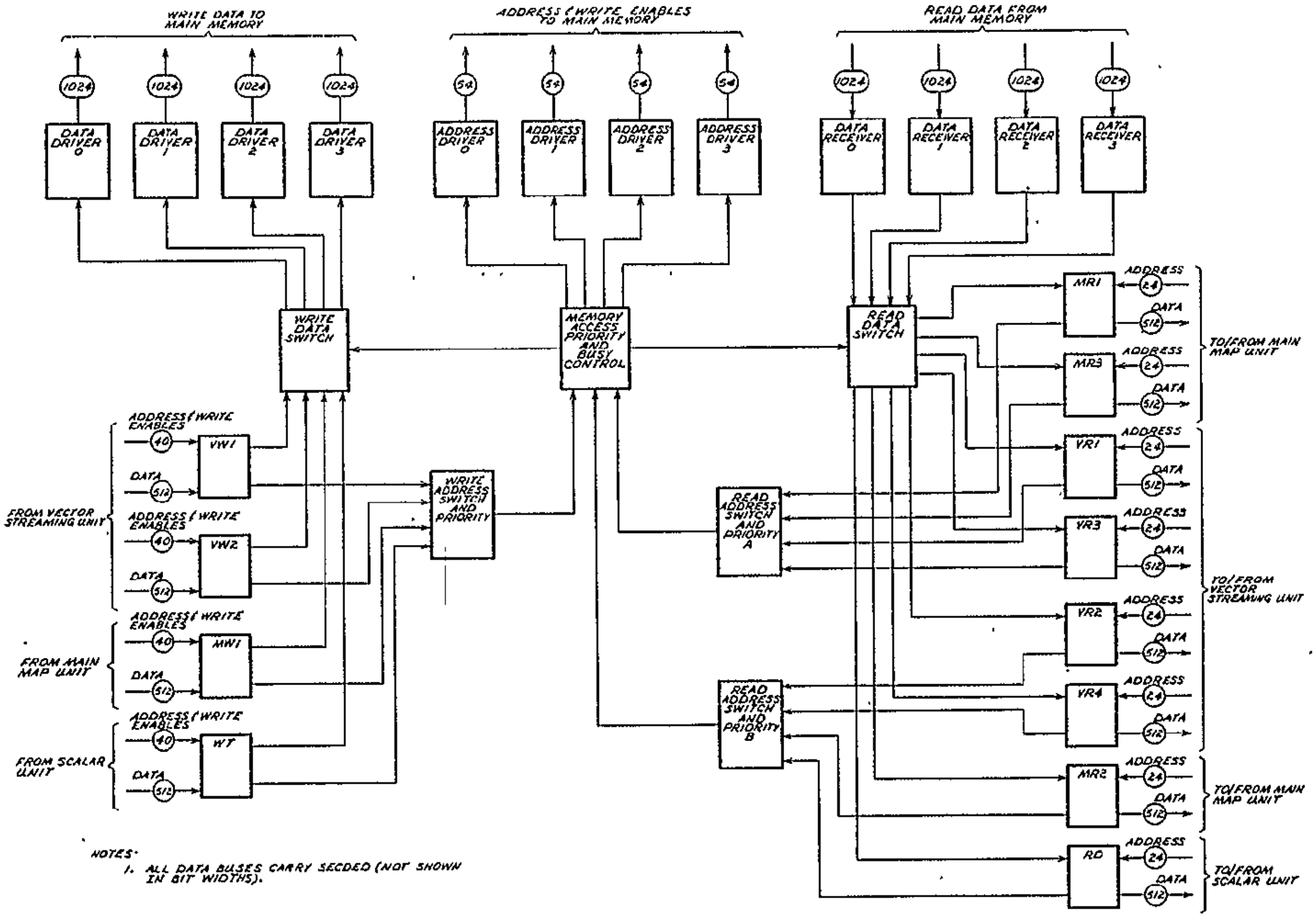
2) Maximize the utilization of the Vector Units--- What if the computations must be done in 64-bit mode and latitude has been exhausted for adding more hardware for parallel processing? If the ensemble could be made to function at near 100% of its capacity (full triadic functions, full time for the whole code), then the 1.2 gigaflop rate would be maintained as the average rate thus achieving the "quintain". It was at this point that the concept of a "tailored" machine for a particular problem environment (wind tunnel flow models) became an aid. Using metrics provided by Ames which were chosen to reflect the desired characteristics, it was possible to determine what hardware emphasis was needed to maximize vector unit utilization. This leads to the next step.

c) Once a memory bandwidth and vector unit capability were established, it became necessary to prove that it could be utilized effectively or the design had to be changed accordingly. Beginning in the first study Control Data had concentrated primarily on the storage capacity of realistically assembled memory systems and the producibility of sufficient arithmetic power. Throughout these early study efforts, Ames personnel cautioned that the data flow required by the large flow model might dominate all other considerations. Using the Vector Units to maximum capacity requires the organization of data into linearly stored vectors of data in memory which can be transferred in groups of elements at each single memory request. Some means had to be found to perform this organization in parallel with computations and without impacting the vector rate.

d) The concept of independent Map Units which could operate concurrently with each other and with the Vector Units arose naturally from this analysis. An initial desire for these units to be as simple as possible had to be compromised with the activities that seemed to be natural to assign to them. If the Vector Units were to be solely concerned with arithmetic processing, then all other memory-to-memory vector operations would have to be performed by the Map Units. The functions to be executed in the Map Units were identified by reviewing STAR-100/CYBER 200 experience with vector processing and examination of the candidate metrics (aerodynamic and weather models).

At first a single Map Unit seemed to be sufficient for the known purposes, moving data only within Main Memory, with all other LEVEL 2 data being transmitted in block form. The split operator method of the implicit code gave rise to easily isolated "transpose" and "slicing" operations that could just as well be performed on the data as it is moved between the LEVEL 1 and LEVEL 2 memories. A pair of seemingly identical units (for programming flexibility) were then specified. Many programming alternatives become evident from this structure, none of which have been explored during this study. An example might be the performing of full transpose operations on meshes and mesh segments in Intermediate Memory while simultaneously performing different transposes on other data in Main Memory. As will be seen later from simulation results, the existing metrics do not fully exploit this concurrency to the degree that seems possible (see section 5).

e) The memory system then becomes the key to the entire design of the FMP, as it must supply continuous streams of data not only to the Vector Units but to the concurrently operating Map Units. This is accomplished by taking the identical memory units used in the CYBER 200 family and increasing the apparent number of memory modules by dividing an existing physical module into four separate accesses. Thus where the CYBER 200 memory can deliver a peak rate of 1024 bits every 20 nanoseconds (actually the memory system can provide data for a range of clock cycles from 10-20 nanoseconds), the FMP system delivers 4096 bits of data (ignoring SECDED) every 16 nanoseconds for a peak bandwidth of 256 billion bits per second. The block diagram shown in figure 14 displays the basic design of the Memory Interchange which manages the data and address streams for the memory system. It is this unit and the data buffering in the Vector and Map Units that guarantee sustained data rates to all elements in the Vector Unit, no matter what the degree of activity is for all components requesting memory. It should be pointed out that the number of LSI panels and cabling for this scheme are considered to be at the maximum allowable by current manufacturing technology. It is felt that no way exists in which this memory system could be extended realistically, and thus no way that a doubling of arithmetic units could be supported with current design and packaging techniques.



NOTES:

1. ALL DATA BUSES CARRY SECDED (NOT SHOWN IN BIT WIDTHS).

Figure 14. Memory Interchange

f) The structure of the FMP as it has passed through its various gestative phases has been tested by estimating the performance of selected segments of the implicit, explicit and weather models which are being used as the analytical metrics for this study. More will be said about this matter in later discussions of the simulation systems for the FMP. The FMP design as it now stands is the culmination of this iterative structuring, testing, and restructuring effort, and does meet the performance requirements of at least the implicit code.

g) Primary emphasis in the analytical effort has always been focused on the three-dimensional implicit model. Only in the last weeks of the study, as CDC engineers had satisfactorily solved the problems in the buildability and RAM features of the FMP, was the full impact of the hardware architecture known with regard to the implicit code. Since that time the FMP design has been evaluated against the remaining three metrics. On a "first try" basis, each of the other metrics have yielded less than the desired 1000 megaflop computational rate. In some cases, the initial mapping of the scalar algorithms did not match the architectural strengths of the FMP. Complete reanalysis and recoding in these cases is called for. Shortcomings in the design have been unearthed also. At some point in the study however, "tinkering" with the design had to end, so that a final report could be completed. Although no design changes are contemplated to optimize the other metric codes at this time, work is continuing on finding appropriate restatement of these codes in the proposed FMP FORTRAN which can approach the 1-gigaflop goal for each metric program.

h) Coupled with all of the above issues has been the problem of programmability, about which more will be stated in the "Software Section" of this report. As hardware features were added or modified in the basic FMP structure, their effect on a possible FORTRAN language system had to be assessed. The continuation of a basic single instruction stream, multiple data stream (SIMD) architecture obviated the need for many FORTRAN language changes, while the ability of a compiler to schedule, stack, and overlap map operations with vector operations impacted the design of the Map Unit and Streaming Control Unit, particularly with regard to the need for and implementation of the "dependency keys".

i) Needless to say the process of steps c) through h) are iterative and will continue in some form even after submission of this report. Once a prototype FMP has been "poured in concrete" as the result of need to freeze design for this report, analysts can spend considerble time learning what the structure really holds in store for programmers, particularly of applications in the areas of the NASF metrics (aerodynamics and weather).

2.3.4 MAINTENANCE

The discussion of first priorities earlier covered the concept of RAM--Reliability, Availability, and Maintainability. The design considerations under that heading dealt primarily with probablistic circumstances affecting RAM, particularly due to the volume of parts and interconnections. Another aspect of the hardware that must be considered in a different light is the marriage of hardware, firmware, software, and procedures to maximize the availability of the FMP. Supporting documents (see Divisions 7 and 8) have been prepared by Control Data specialists as "position papers" on the maintenance strategy and diagnostic strategy that should be pursued for FMP-scale systems.

A cornerstone of FMP availability is the ability for the system to recover, diagnose, isolate, and be repaired in the minimum interval of time lost to the consumer. In addition to the issues discussed in the position papers, certain aspects of the FMP impel offering some supplementary commentary.

a) The maintenance function as a hardware concept---

The magnitude of the NASF makes complex interactions of its constituents inevitable. The quantity and complexity of the elemental relationships in this system further ensure that even trained personnel will find the evaluation of many system failures difficult or nearly impossible in the brief time allowable for interruptions during operational hours. Computerized assistance is mandated in this situation, however none of the potentially failing computers in the system can be expected to diagnose itself. A separate computer could be given this task, but if it becomes itself debilitated then it could become more of a debit than of value to the overall availability of the system as a useful resource. The need then is for an abstract concept that can be fleshed out, and farmed out to the appropriate programmable processors in the system. The reason for discussing the maintenance function in the abstract is that it first can be created and described as if it were a centralized function, despite the fact that in actual implementation it is distributed as widely as the functions it is trying to manage.

Philosophy--Each programmable entity in a system must possess means to disclose to the outside world the existence and nature of any failure that might occur, even when the flaw is correctable. A failing entity should not be required to diagnose itself, let alone cure itself, but should provide the maximum information about itself as possible, as would a patient in a clinic, to speed the diagnosis and treatment process. The maintenance function, like a concientious physician or health service, must collect, collate, and analyze all physical symptoms in health as well as for illness. For like the physician to whom every clue and pattern could be significant, the contemporary system practitioner needs every allusion in pursuit of quick repair. Diagnosis and collection of maintenance clues must be included in the normal workload of each system component.

Implementation--From the outset of design all hardware and software components must become subscribers to the philosophy. "Hooks" must be built into all elements of the system where engineering analysis indicates potential need and practicability. What does this mean to the FMP?

First some means of testing, measuring, and sampling the activity of critical networks must be engineered into all critical networks of the system. This implies that some systematic study should be done to identify the critical networks. In the FMP some obvious places to put the diagnostic "stethoscope" come to mind. The SECDED checking networks not only must supply error indicators, but also should provide "Polaroid" glimpses of the failing data segment and associated addresses to the maintenance monitoring system. The comparator networks in the Vector Units must also feed failure information. Not only an error flag should be recorded, but the actual data mismatch must be frozen and registered in the error logging of the maintenance system.

In the FMP, SECDED check bits are carried as far along data trunks as possible so that the trunks and intervening paths can be corrected, as well as the memory. This does not mean that the SECDED checking networks need be limited to the trunk ends, as isolation then becomes more difficult. Instead, the checking networks can be placed at strategic locations in the system. The current design calls for placement of SECDED checking in all hardware blocks where the symbol SECDED appears in figure 15. SECDED correction, however is limited to the ends of the trunks only.

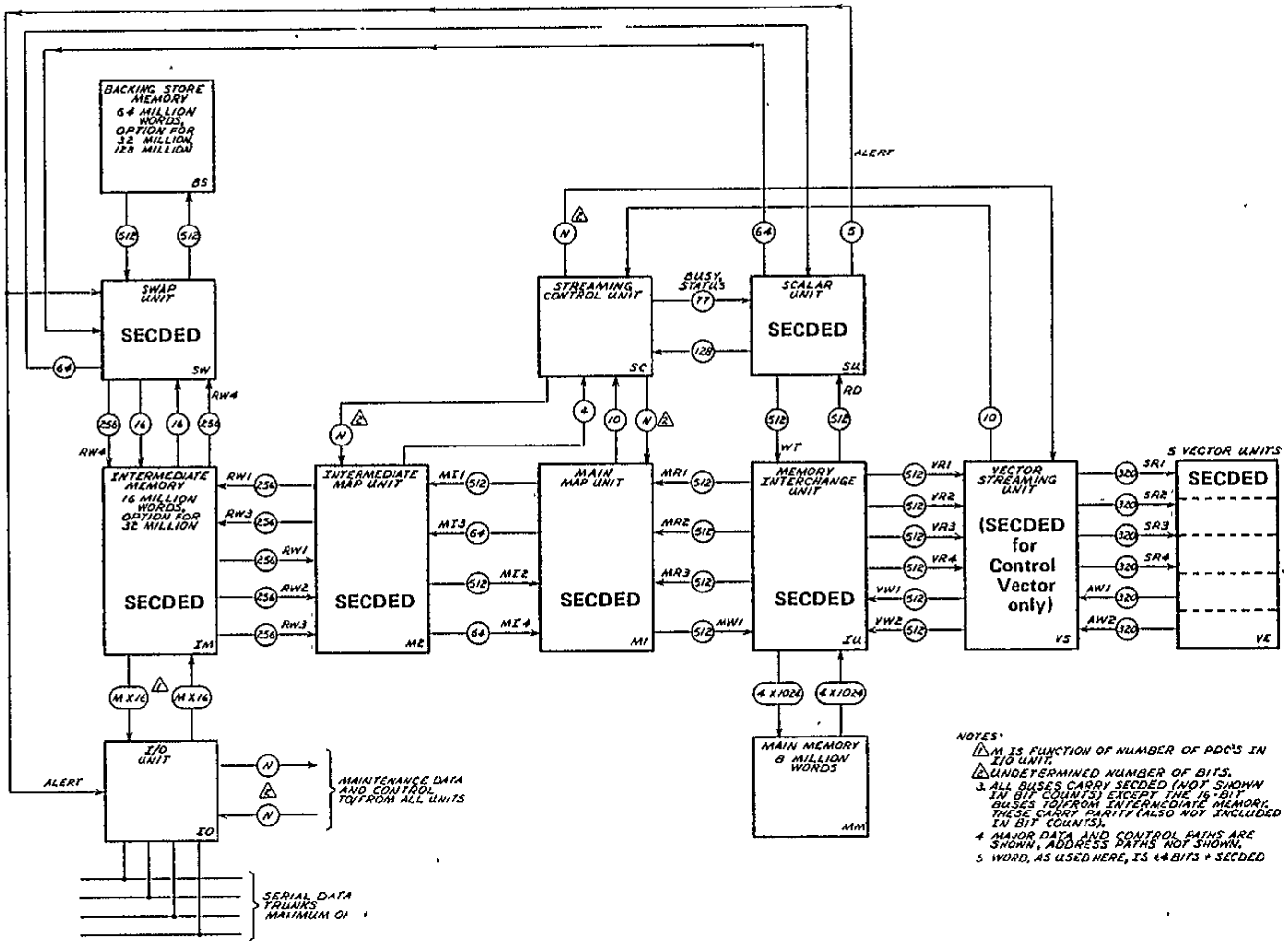


Figure 15. SECDED in the FMP

As can be seen in the block diagram of figure 15, Main Memory SECDED will be checked at the Memory Interchange as the data is transmitted to the Vector Unit. The SECDED is again checked within the Vector Unit, and any correction applied there, if needed. The major difference between these two SECDED networks is that since the first network does not actually modify bits in the data stream to perform correction, it can be an "out-of-line" circuitry which does not add time to the data path. The potential correction of data in the Vector Unit, however requires logic to be interposed in the data path at a consequent cost of several nanoseconds in transmission time. The isolation aid that the additional SECDED checking gives should be obvious from the diagram. A SECDED error reported by all port checking units in the Memory Interchange, for example, would point to a basic data failure in memory. A single port SECDED failure would indicate a problem in that particular port's trunk hardware. An error reported only by the Vector Unit SECDED network indicates a failure in the data trunk or the paths in the Vector Streaming Unit (which include high speed shift networks).

This simple example illustrates the complex combinations of symptoms that need to be evaluated in light of the architecture to arrive at a specific failure point where the maintenance engineer can begin his search. A computerized correlation and display of the possible failure points would surely speed up the isolation process.

In addition to the SECDED and Vector Unit comparators, certain second order error detection and reporting schemes are employed in the FMP.

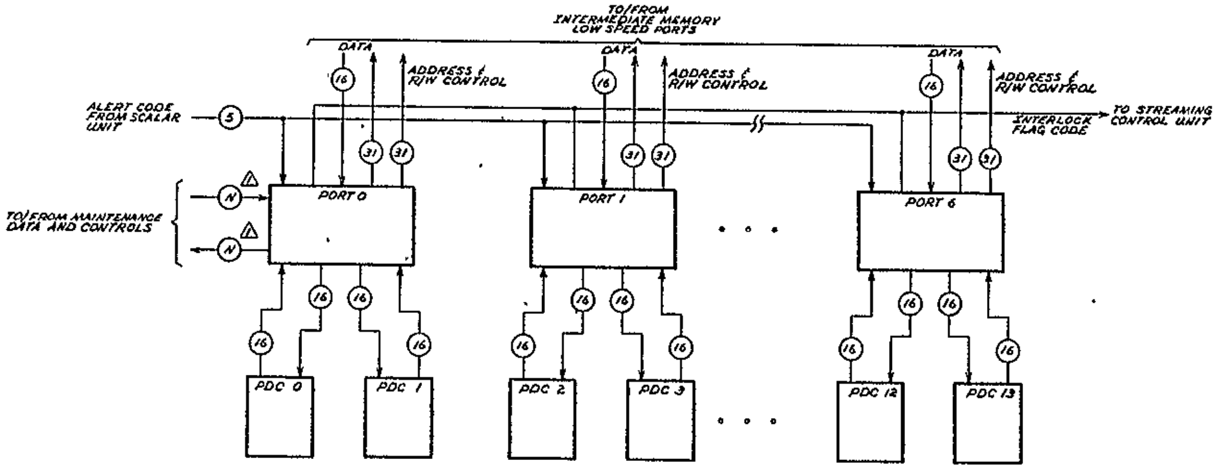
- Illegal function detection---Each of the independent units possesses fewer operating modes than the function codes transmitted to them are capable of encoding. For example, not all of the 256 potential operations available in the main instruction stream are legal. Thus all illegal functions, when detected, bring the FMP to a halt and send a flag to the Maintenance Control Unit (MCU).

- Time-out detection---Certain of the FMP units expect a maximum allowable time for response to function and memory access requests. If the maximum time is exceeded the FMP is halted and an error flag sent to the MCU.

- Monitoring counters---Counters internal to the FMP can sample various activities of the FMP, including the running clock, and can transmit the accumulated data to the MCU on a regular basis. In addition to the obvious utility in performance measurement and optimization, certain measurements may provide a clue to incipient failures in the system. For example, if a vector operation of length 1000 elements consistently runs slower than some nominal value, there may be a component failure in the memory request scheduling logic of the Vector Streaming Unit. Note that this failure is subtle, since all answers may be correct, but the program runs slower than the hardware design would normally permit.

- Self checking---Some networks may have a built-in ability to generate and check simple test cases during idle periods. For example, an address adder in the Vector Streaming Unit might have alternative input selects that permit the injection of all "zeroes" and all "ones" patterns with a built-in check for the correct carry, generate and overflow bits. Since the adders are highly underutilized many idle cycles would be available for such checking.

How does the FMP hardware gather all of this data from its various limbs and actually notify the appropriate maintenance function? Each separate network in the FMP that generates error data, transmits this data to the I/O Unit where special hardware in I/O PORT 0 (see figure 16) locks up the information in static registers and sets a flag indicating that maintenance data is present. Any PDC (programmable device controller) connected to this port can-and will sample this flag and related registers, forming a message directed to the "maintenance function" (note this is the abstract of a specific hardware that used to be called the "maintenance unit"). Maintenance function messages are put on the network trunk where they are transmitted to all other attached processors. Any processor which recognizes itself as an "addressee" for maintenance functions will read the entire message and turn over the contents to stored programs which will analyze the maintenance information given.



NOTES.
1. UNKNOWN NUMBER OF BITS.
2. PARITY CARRIED ON ALL 16 BIT DATA LINES (NOT SHOWN).
3. ONLY PDC'S 0 AND 1 CAN FUNCTION AS THE MCU INTERFACE.
4. THERE MAY BE A MAXIMUM OF 7 PORTS AND 14 PDC'S. (THE BASIC SYSTEM WILL HAVE 3 PORTS AND 6 PDC'S)
5. NO DROPS TO THE SERIAL TRUNK SYSTEM ARE SHOWN BECAUSE THE NUMBER OF DROPS FROM EACH PDC IS DETERMINED BY THE SYSTEM CONFIGURATION AND RELIABILITY REQUIREMENTS. (4 DROPS/PDC MAXIMUM).

Figure 16. FMP I/O Unit

The potential exists then for several computers on the network to possess all or part of the maintenance function software. Which computer provides the specific maintenance service depends on which processor has its "turn in the barrel", or is available to assist the FMP. In other circumstances the service may be performed by the lowest level processor that can handle the task. For example, a small miniprocessor such as the CYBER 18 might be attached to the trunk for system statistics gathering and initialization purposes. A correctable SECDED error might be reported on the trunk, and this small processor given the responsibility for collecting the data and storing it on disk for later analysis. A total failure of one Vector Unit, however, when reported by messages on the network trunk, might require the use of one of the major front-end processors to determine what strategy is to be followed (launch immediate repairs since the workload is light, swap out the unit and perform automatic on-line diagnostics, etc. ...). The content and not the addressee of maintenance messages determine which of the system resources will perform the maintenance action. In some cases the small CYBER 18 might be off-line for maintenance, and thus the SECDED error recording must be handled by another processor in the network. The system must not allow the information to be lost. Returning to the anthropomorphic simile of the physician and patient, the FMP puts out a cry for help and the most competent individual capable of rendering first aid attends the victim.

In like manner all other components of the NASF place messages on the trunks to provide operational and error information to be received and processed by the maintenance function. The PDCs themselves place error and operating data messages on their own trunks. The support processor system hardware may not provide specific electronic linkages to the network and attached maintenance functions, however it should be clear that maintenance function messages can be generated by operating system software as well as by hardware.

The maintenance function need not, and must not, be purely passive in nature. Those elements charged with maintenance function responsibilities must be able to deadstart, restart, and recover any system component by transmitting special messages on the trunk. The maintenance entity must be able to make decisions about resource allocation and turn trunks, processors, peripheral devices, and remote accesses on or off as system failures make such actions necessary. Certain diagnostic sequences will have to be initiated, and oftimes monitored by the maintenance function.

b) The maintenance function as a software concept---

In the previous discussion on hardware there are many implied objectives for the designer. Some electronic "hook" must be designed into new components so that maintenance functions transmitted on the network can affect some hardware operation, as for example the concept of "SYSTEM MASTER CLEAR". The hardware hooks, however, can only provide information which must be collated and analyzed. The initiation of maintenance actions and the evaluation of maintenance data must be done by one or more software systems. Error data can be created by software also. The maintenance strategy must consider the relationship of software to the maintenance hardware system. Some points to be included in such ruminations:

- All maintenance software, with the exception of network trunk interface firmware, must be "transportable". This means that any one or any group of maintenance functions can be "farmed out" to any of the processors in the NASF. Transportability implies the use of a commonly available higher-level language that would be supported on all programmable processors in the system. The growing universality of PASCAL as a systems programming language makes this the prime candidate for maintenance software. In addition to the language, the structure of all maintenance software must be carefully specified, tightly controlled, and implemented in a disciplined manner. In specific terms this means that a small diagnostic monitor with standard software interfaces must be created in such a way that it can fit in the smallest processor that might be assigned maintenance functions. The standard interfaces necessary would consist of common message formats for communicating with the outside world (via the network trunk), a common set of diagnostic primitives (or calls to kernel subroutines), and a common set of definitions of data and procedures as well as common data descriptions which would be incorporated as declarations in each PASCAL program.

- All operating system software should provide maintenance oriented messages in the same manner as the hardware in the FMP and network trunk. These messages should provide normal operating information (such as "time-stamped" job start/stop messages) as well as warnings of exceptional conditions. Examples of this latter message traffic would be: abnormal job termination, failure to receive responses from other processors in the system, impossible conditions detected within the operating system such as table overflows, missing pointers, etc. The concept of collecting, recording, and disseminating maintenance data to other processors should be fundamental in the design and implementation of all operating system components and not just merely a set of hooks to emplace in developing systems.

- Critical maintenance functions such as system shut-down, restart, and recovery must be provided in at least two different processors with appropriate interlocks so that only one of the pair will respond to emergency conditions.

- The maintenance software must provide additional security safeguards for access to maintenance data and for permission to invoke maintenance software functions. A powerful tool in the maintenance arsenal will be the remote access (via standard interactive terminals) to the FMP by specialists not necessarily resident at the NASF site. This can only be permitted if adequate protection is afforded the system from illicit use of the maintenance capabilities. The level of security must be greater than that which is nominally acceptable for commercially available computer based systems. The reason is obvious, since remote access could be obtained to functions which affect the actual hardware operation. Certainly the software system must be able to prevent catastrophic events from being initiated while the system is in production mode.

c) The maintenance function as a procedural concept---

In Division 7 the subject of maintenance strategy is discussed. In addition to the hardware and software that are designed into the NASF, a procedural strategy which concerns itself with manpower (numbers and qualifications), organization, economics, methodologies, and tradeoffs in the maintenance process must be designed as well. One of the major operating costs of the NASF will be that of the maintenance support function. Not only must a sufficient number of trained personnel be available for emergency service, but personnel, equipment, and parts must be on site to perform regular preventative maintenance. The costs of maintaining the maximum work force and parts inventory can be prohibitive and must be weighed against the alternative probabilities of system failures during time of operational use.

2.3.5 MICROCODE CONTROL AND FAULT ISOLATION

A powerful tool for computer designers, made possible by advances in memory technology, is the microcode control of logic networks. Although quite often stored in ROM (Read-Only-Memory), microcode can also be stored in standard RAM (Random Access Memory) and loaded into the CPU at every system "deadstart" (or power-up, or "coldstart"). The use of reloadable microcode permits the incorporation of design improvements in the control systems without changing components or rewiring logic in the computer at a field installation. A class of memory technology is now available that permits the use of microcode control for even the highest performance logic family, with a minimum of auxiliary hardwired control necessary to meet the speed requirements of most networks.

The Control Data FMP design is based on the use of this high performance microcode, which is distributed within each functional element rather than being centralized in a main control unit. In addition to the concept of "distributed microcode", the machine employs a "two-dimensional microcode" structure which is utilized for on-line diagnosis and fault isolation. Figure 17 sketches the concept as it appears to the hardware designers.

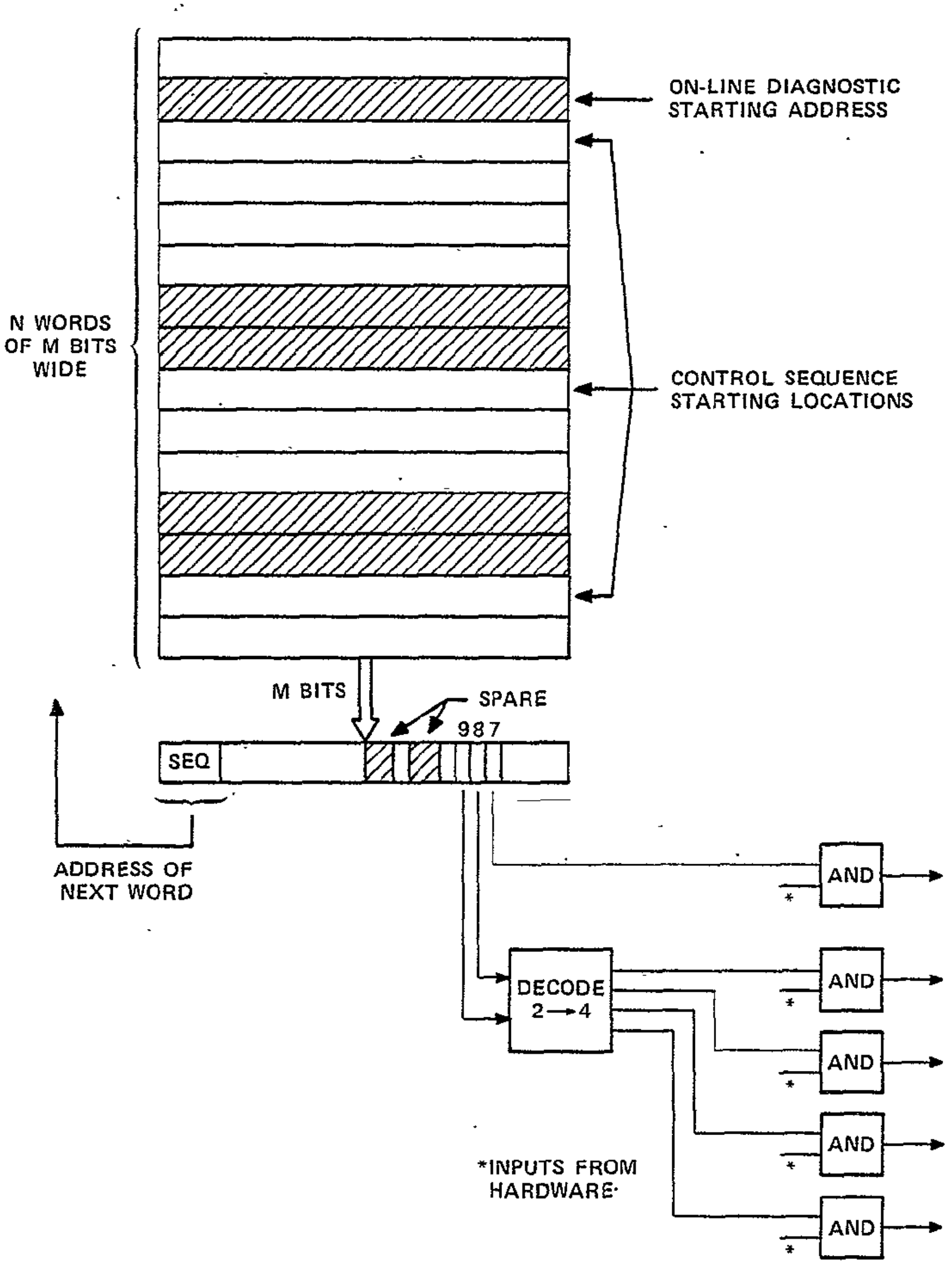


Figure 17. Microcode Memory

The microcode memory consists of a semiconductor system capable of storing N words of M bits in a high speed RAM technology package. Control is achieved by reading a word from this memory and distributing the bits to actual hardware gates in the CPU. In the example shown, bits m through n are used to control the next address to be read from the memory. Bit 7 is shown being sent directly to a hardware AND gate which controls some data path. Bits 8 and 9 are sent to a hardware 2-to-4 decode network which converts the two-bit code into four distinct control signals which are then transmitted to data path controls elsewhere in the network.

A typical modus operandi for such a microcode is for it to receive an initial starting address from an outside source (say for example, a function code being sent from the Scalar Unit to the Vector Unit). This address points to one of three legal starting addresses in the microcode memory. The first control word is then read from that memory and its bits disseminated to the appropriate hardware. The next word to be read is then determined from the SEQ field of the first word, and the process continues until either the microcode shuts itself off and awaits a new external address, or perhaps, enters an "idle loop" (sequence of do-nothing microcode instructions).

Since memory systems are usually packaged in convenient sized units (for example, a single efficient block might be 32 words of 32 bits), the designer usually finds that after he has completed his control structuring there are unused memory locations and unused bits in each distributed memory (crosshatched areas in figure 17). The two dimensions of this microcode are evident from the diagram -- depth (number of words) and width (number of bits). The realistic designer will first set aside some of the spare depth and width for possible later design improvements. However, the need for reliability and availability dictate that a certain amount of words and bits be allocated to the maintenance function. Some of the bits might be connected to gating networks which are not used during normal operation. Others might be sent as codes to the Maintenance Control Unit as error flags.

The spare words in the memory might be employed in two ways:

- First the idle loop, which the microcode normally enters while awaiting a new function, might be extended to use these words to initiate self-checking actions in some of the networks. An example previously given was the forced generation of all zeroes to an adder network. At the same time, one of the spare bits might enable an all zeroes check to be performed at the output of the adder. If this check is not satisfied an error flag could be sent to the MCU.

- Second, an external source (such as the MCU could initiate a microcode sequence at one of the spare words (labeled ON-LINE DIAGNOSTIC STARTING ADDRESS in the figure). This sequence could perform certain unique network control not available in normal operation. An example of this would be the defeating (shutting off) of the generated carries in an adder network so that only the partial sums are transmitted to the output trunks. In some cases this might aid isolation of failures a great deal. Once the sequence is completed the microcode might then return to its normal idle loop.

A third diagnostic and isolation aid is the ability to load a completely different microcode into the memory from the maintenance station (this function is called "down-loading"; or "downstream loading"). Down-loading of the normal operational microcode is always done at system startup time, since the RAM memory is volatile. It can also be done independently to any independent functional element in an off-line manner by the maintenance station. This version of the microcode would be called the fault isolation microcode. It may consist of one or several loads of microcode which trigger and test the logic networks at the gate level. The intent of this system is to provide back to the maintenance station sufficient analysis to permit the determination of fault location to the smallest replaceable component level. Loading of such microcode into one Vector Unit does not affect any other unit, therefore it is possible for the MCU to switch into operation the spare Vector Unit, load isolation microcode into the failing Vector Unit and perform isolation exercises in that unit while the remainder of the FMP continues normal operation.

It should be obvious that the designer must design fault isolation into the FMP from the start, and not just be content with using whatever spare words and bits may abound. The design philosophy for the FMP has been to add whatever extra bits and words are necessary to make achievement of the replaceable component isolation goal a reality, within the constraints that too many words and too many bits may take up too much room or too much access time as to be feasible.

2.3.6 VALIDITY AND VERACITY

The primary objective of this study has been to arrive at a system design which can be built and utilized effectively within a very tight development time schedule. The willingness of NASA and manufacturers such as Control Data to launch into the actual construction of such a facility with all its visibility and all its risks will be determined not only by allegations of achievable goals, but more importantly by the confidence upon which all parties can rely on the cost, performance, schedule and RAM goals.

A summary of statements regarding the validity or correctness of the conclusions on hardware design as far as is known today are presented here.

2.3.6.1 ASSUMPTIONS

1. Design would not commence until mid-1980 at the earliest.
2. A FORTRAN compiler can be produced which can generate and schedule machine instructions from the implicit metric as shown in appendices B and C.
3. The entire system design and manufacture would be conducted by a single vendor, with choice of new equipments installed with this system left to that vendor.
4. The implicit metric reflects the most probable performance characteristics of the major load of production jobs on the FMP.
5. The system load data provided by Ames is at least within 20% of reflecting the 1984-1989 production environment.
6. Installation and operation of the NASF can be phased over a period of 12 months as more and larger elements are needed to support the increasing workload (for example, the Backing Store need not be installed at the outset to meet initial program development and production operation. Then it could be installed in increments.)
7. Sufficient funding and compute power will be available for the duration of the development project to support extensive simulation of the FMP and the system at all levels (system, block, and gate-level models).

2.3.6.2 VALIDITY DERIVATION

1. All circuit components costs, failure modes, quantities, and complexity are based on existing parts now in production with the exception of the Backing Store technology.
2. The Backing Store estimates are based on MOS technologies now in production, and scaled for the more complex chips in that memory.
3. Where sufficient production volumes have not yielded sufficient experiential data (as in the case of the LSI-168 gate array and Intermediate Memory chip) conservative learning curves for failures and costs were utilized.

4. The packaging, power, and cooling technologies were taken directly from computer products already in existence.

5. Gate-level design of some of the key FMP elements, or portions of those elements (such as the Memory, PDC, Scalar Unit), have been completed and either put into practice or simulated in detail as part of standard product development on other CDC projects.

6. Design and simulation techniques are identical to those employed for other large scale computer development efforts.

7. To the extent that time and resources have permitted, the design of the FMP has been carried to the point where all manufacturing and performance aspects are expected to be within 10%.

8. All other system components in the NASF are taken from the warehouse of existing system devices, for purposes of calibrating cost, performance, and physical requirements. Since forthcoming products will undoubtedly improve with respect to all three parameters, the current estimates are considered highly reliable and quite conservative.

Finally, as a bottom line, based on two decades of effort to build and support machines of this class, Control Data believes that if the requirements reflect reality and the system design truly meets the performance objectives of the projected production environment, then the entire NASF is producible in the specified time frame and will be effective in its given role.

3.0 SOFTWARE DESIGN

The success of the FMP rests not only on the ability of the hardware to attain its performance goals, but also on the effectivity of the software system that supports the hardware. The first concern is that the problem statement be matched to the computer architecture to maximum effect. This is a function of the programming language characteristics as well as the capability of the compiler to make automatic decisions, transparent to the programmer, regarding the scheduling and optimization of the FMP hardware resources. The major effort in this study has been to attack these two aspects of FMP software design.

The overall FMP capability for initiating new jobs and scheduling system interactions such as I/O with a minimum of lost time to the production jobs is a second software design concern which has been addressed in these three NASF studies. This report will discuss the current conclusions on these issues, and will include updates of previous study findings in these areas.

3.1 LANGUAGE ANALYSIS AND DEFINITION

At the outset of the first NASF study Control Data concluded that a special purpose language would have to be developed for the FMP to ensure that the best match was made between language and problem algorithm, and between language and machine architecture. The pragmatics of the expected operating environment made it impossible to ignore the imperative of maintaining a FORTRAN-like language for most, if not all, applications programming. Language design for the FMP would focus entirely on the applications development aspect of program statement, leaving the operating system and support software development in the hands of the extant system language "fad".

3.1.1 EVOLUTION LEADING TO SPECIFICATION

Two rationale dominate the CDC recommendation for a special language for the FMP.

1. The possibility that compiler technology will develop "production" language processors that can unravel the existing scalar FORTRAN metrics into sequences suitable for parallel computational elements is an exciting object to pursue, but the risk of failure at this stage is quite high. This alternative must not be abandoned by researchers however.
2. In the long run, applications programmers will benefit from conceiving of their problem solutions in parallel form, rather than letting a "smart" compiler invisibly

transform their code. By making the inherent parallelism visible, the programmer can assist the compiler in the production of optimum machine instruction sequences.

Demonstrating RADL researchers resolve and technical opinions, at least five different approaches were attempted at defining a programming language which met the criteria of compilability, consistency, clarity, and most of all, acceptability to the potential user community (NASA-Ames). The approaches began in the initial study period by beginning with a single language construct to FORTRAN and thence experimenting with the resulting language using the implicit and explicit metrics. As shortcomings were encountered in use, an additional construct was created and added to the language specification. In that manner, a draft language enhancement to FORTRAN was developed during most of the first study. In the main, these extensions were reasonably acceptable to those programmers who considered them, however they did not meet the criteria of consistency and compilability, appearing much like a set of patches.

A quick remedy for this was to reduce the extensions to a manageable size and ignore some of the picky programming details (such as data movement) that arose; so evolved the CODO (concurrent DO) constructs described in reference 1. As the second study began, it became obvious that other considerations might weight heavily on decisions regarding language.

1. The expected system development time for the NASF would severely strain the best efforts of any compiler development team. To reduce development risks an existing compiler would have to be adapted to the FMP usage. Otherwise, the first years of NASF production would be fraught with the problems inherent in any new, complex compiler system.

2. The acquisition of "parallel programming" insight is painful on first go for even the most highly skilled analysts. The process of learning new thought processes, researching and developing new parallel algorithms is, in its initial stages, quite lengthy and requires great diligence.

3. STAR-100 experience was beginning to show the direction that algorithms could take, and an inventory of useful programs, subprograms and functions was being developed that might be of great value, in their most mature forms by 1984, to a widely used system such as the NASF. If some means could be found to utilize all of this background derived from the STAR experience, much time and effort might be saved, hence costs and risks might be reduced.

This thinking led down the STAR-100 byway, in an attempt to directly apply the STAR-100 FORTRAN language extensions to FMP programming. The obvious advantage of this was that programs could be coded, reformatted, debugged, and even small production runs made on existing computing systems. The disadvantage was that although the language was consistent and obviously compilable, the user acceptance was near zero. The major objections by NASA were:

1. The need for explicitly describing vector data structure and vector operations made it difficult to develop algorithms without being intimately familiar with the internal hardware design. For three-dimensional simulation problems this was an undesirable requirement, since most creative resources are expected to be needed in just solving the physics and mathematics of the model solutions.

2. Some changes were needed in the STAR-100 FORTRAN to accommodate the differences in architecture between the FMP and STAR. This meant that the compiler could not be retained in its original form in any case.

3. Some notational forms were clumsy, but required by the compiler. There were too many additional constructs to learn besides the normal FORTRAN constructs.

4. The existing metric code would have to be severely disrupted to organize the code in a form suitable for optimum machine operation using the extensions.

Control Data was asked to go back to "square one" and attempt to provide a FORTRAN system where most of, if not all, the machine architecture could be "hidden from view" of the programmer, relieving him to deal with the mathematical and physical intricacies of his problem. Interaction with Ames programmers revealed the fact that they objected to the explicit knowledge and manipulation of Vectors of the STAR form but had no reservations about dealing with an abstract collection of data called an array, or a subarray, as an entity in an single computation. Thus the programmers claimed they would be happy with the ability to state solutions like:

```
PROGRAM TEST

DIMENSION A(100,100,100),B(100,100,100)

A=B*B
```

where the entire meshes A and B take part in the one operation. Further, the manipulation of subsets or subarrays of arrays was also acceptable, as long as the compiler worried about partitioning the actual arithmetic operations into sequences of suitable vector operations for the FMP. Thus:

```
A(1:10,1,1)=B(1:10,1,1)
```

which causes the first column of B (ten elements) to be moved to the first column of the array A.

It is easily apparent that this same operation could be restarted as a conventional FORTRAN DO loop:

```
       DO 10 I=1,10

   10  A(I,1,1)=B(I,1,1)
```

but the clearly recognized movement of a column of data in the subarray form describes the desired action, while statement 10 above might be submerged in a morass of statements in a much larger DO loop in I. The clarity of intent is not only more open to the casual reader but to the compiler itself which has to deal with messy DO constructs containing IF-ELSE blocks, GOTO and CALL statements which may make the simple array move operation impossible to determine when stated in DO loop form.

Given this assistance and guidance, RADL set about reconciling those desires and previously stated constraints with another new set of considerations:

1. The final FMP form arrived at in this study deliberately separated the data movement (map) operations from arithmetic operations in terms of hardware and control. Some of the resulting concurrency could be discerned automatically by the compiler, but a modest amount of language assistance might make it possible to fully utilize the concurrency and maximize vector unit activity.

2. Tradeoffs in FMP architecture resulted in a three-level memory hierarchy which cannot be entirely hidden from the programmer due to the vast differences in performance levels and constraints on data storage (only BLOCK transfers are permitted to/from LEVEL 3, for example).

3. The ANSI FORTRAN 77 specification was finally adopted in final form and commitments to implement this language as the new US standard were practically imposed on American computer manufacturers. Hence all CDC FORTRAN compilers were rapidly to turn into ANSI 77 processors.

4. Some of the non-vector extensions of the STAR-100 compiler proved to be quite valuable in programming for the STAR and CYBER 200 machines. The subarray reference notation, IMPLICIT, CHARACTER, and extended intrinsic functions have demonstrated high utility in production codes.

5. The need to base the FMP compiler on an existing compiler is still considered imperative to meet the NASF development schedule. This is particularly true if the language construct additions or changes can be minimized.

The outcome of these deliberations is offered in Volume III, FMP FORTRAN Language Specification. A more detailed discussion and demonstration of the language facilities is included in subsequent sections of this report.

3.1.2 OBSERVATIONS AND RATIONALE

The preceding discussion has concerned the evolution of thought and deed that led to the language specification given with this report; however, some additional commentary should be offered.

1. Programs and algorithms developed for the STAR/CYBER 200 family can be directly transferred to FMP FORTRAN by converting the explicit vectors to subarrays. This can be done easily with a mechanical "SIFT" (conversion) program. STAR "Descriptors" can be directly converted to the more flexible DYNAMIC variable.

2. Dynamic assignment of memory levels using the DEFINE statement makes it possible to use the identical production code for small problems (which would fit entirely in Main Memory and operate quite efficiently) and large codes (which need to be based in Intermediate Memory, with the slow map times thus incurred), without recompiling. The programmer may, if he wishes, take direct control of temporary space allocation for large arrays.

3. Scalar algorithms may be vectorized directly in many cases by redefining all scalars as arrays or subarrays with DYNAMIC statements.

4. In the examples in the implicit code report the full subarray reference notation is used. In many instances this would be unnecessary once the compiler could detect these cases from the normal array notation imbedded in a DO loop. The subarray references were retained since it was felt that they more clearly describe "what's going on".

5. Some form of dynamic redefinition of data structures is necessary in parallel systems to avoid being bound by the static characteristics imposed by DIMENSION which makes it difficult to store adjacent elements together, column-for-column and plane-for-plane, (This is, incidentally, an opinion stated by Dick McHugh of CDC in 1976, but not put into practice until now.)

6. Although compilers can now allegedly process and incorporate more than a single module at a time, and thus vectorize across CALL statements, the use of a single generalized subroutine (with its inevitable IF statements used to pick out special cases) is basically inimical to automatic, OPTIMUM vectorization (notice the emphasis on OPTIMUM!). The recoding of STEP brought the XXM, YYM, and ZZM subroutines in-line, not only to aid vectorization but to eliminate redundant data transfers (for example the use of the array RJ, lines 930, 1030, 1040 of appendix B). In a similar vein, VISMAT needs to be brought in-line so that the subroutine call to ZZM can be eliminated and the data previously computed reused.

3.2 FMP LANGUAGE DESCRIPTION

The base language chosen by Control Data for FMP applications programming is ANSI FORTRAN 77. A set of extensions to this basic language has been defined for the CYBER 200 family, and these extensions have been further augmented by specific FMP features intended to assist the compiler in producing optimum code for the FMP. The choice of ANSI FORTRAN 77 was based on the following considerations:

a) The FORTRAN language, imperfect as it may be, is commonly known, and has become the de facto "lingua franca" of the American computer community. It can be expected that all potential front-end or support processors will supply FORTRAN compilers as part of their standard software system. Thus most applications programming outside the FMP will most likely be done in FORTRAN.

b) The ANSI 77 version of FORTRAN will shortly become the official standard language specified in all government procurements (and thus quickly be required in all commercial computer purchases). Absolute requirements for vendor supply of ANSI FORTRAN 77 compilers are estimated to be imposed in the middle of 1980. It is expected that after some interim period following the introduction of the ANSI 77 requirement this updated FORTRAN will become the sole FORTRAN language supported as "standard" by the offerers of NASF processors other than the special purpose FMP.

Volume III of this report contains a specification of the proposed ANSI FORTRAN 77 language, as amended for the FMP, in the form of a programmers' reference manual. The specification consists of an original CYBER 200 (STAR) FORTRAN manual, with line and page changes incorporated as insertions to the original. This was done to make visible to potential users the differences between ANSI 66 (as represented by the CYBER 200 FORTRAN) and ANSI 77, as well as the differences between CYBER 200 FORTRAN vector features and FMP vectorization aids.

The language specification can be viewed as four distinct entities:

1. The basic reference manual.
2. Changes needed to make the language fully ANSI 77 compatible.
3. CYBER 200 FORTRAN 77 additions (beyond ANSI 77).
4. FMP FORTRAN extensions.

This language, as described in Volume III, was used for the recoding of the implicit and explicit aerodynamic flow codes as presented in this report.

3.2.1 THE BASE DOCUMENT

The reference manual (Volume III) provides a FORTRAN language originally based on ANSI FORTRAN 66. Extensions to permit implicit and explicit vectorization for the STAR computer were added to the language. Data types such as BIT and CHARACTER were included to support the string processing capabilities of the STAR architecture. Additional vector constructs were added, such as DESCRIPTOR and DOUBLE DESCRIPTOR, to provide a "shorthand" means of invoking standard and sparse vector operations. The explicit vector constructs that were added for STAR have been replaced by more flexible constructs for the FMP version of the FORTRAN language.

The majority of the dialect described in this base document is familiar to all FORTRAN programmers. One construct that was added to STAR FORTRAN has proven to be quite useful in the analysis and programming of FMP codes, and thus deserves some discussion here; that construct is SUBARRAY notation, and rationale for it follows.

In the early developmental days of the first "vector" or "parallel" machines (ILLIAC IV, STAR, TI-ASC, and PEPE) compiler developments discovered that automatic vectorization of FORTRAN code was made difficult by common programming practices that incurred no penalties on scalar computers. Such practices could be represented by a DO loop of the form

```
     DO 10 I=1,1000
     A(I)=B(I)*C(I)
     .
     .
     CALL GLUMPF(A(I),B(I))
     .
10   A(I)=A(I)**2
```

If the CALL statement were not present, even primitive compilers could automatically create vector operations for the arithmetic assignment statements shown. The introduction of discontinuities such as this, when all program modules cannot be compiled together, makes vectorization nearly impossible. If, in fact, the arithmetic could be vectorized, the programmer could restate his intentions as

```
      DO 100 I=1,1000
  100 A(I)=B(I)*C(I)
      DO 101 I=1,1000
  101 CALL GLUMPF(A(I),B(I))
      DO 102 I=1,1000
  102 A(I)=A(I)**2
```

The compiler is thus assisted in its vectorization task by the programmer restating the data flow more explicitly, although a bit ponderously. To get optimum results on the new generations of vector machines, it has become necessary to make the programmer conscious of the "parallel" or "vector" nature of his programs so that problem restatements can be done intelligently. The consciousness must also extend to the data allocation schemes, since some processors possess memory hierarchies with significant performance differentials at each memory level, while other processors require linear, sequential storage of data for optimal vector performance. Additionally, some form of "shorthand" would be desirable to reduce the coding effort while improving readability.

Early in the 1970s, several different computer developers proposed a set of extensions to the ANSI FORTRAN committee for the description of operations on whole arrays or portions of arrays called "subarrays". These recommendations, while not approved by the committee, appeared to have sufficient support that they were implemented in at least four different vendors' compilers, one of those being the STAR FORTRAN 66 compiler.

Basically, subarray notation can be thought of as another means for describing processing that is commonly assigned to DO loops. The simplest case is to perform operations over an entire array:

```
     DIMENSION A(100,100,100),B(100,100,100),C(100,100,100)
     .
     DO 10 I=1,100
     DO 10 J=1,100
     DO 10 K=1,100
     A(I,J,K)=B(I,J,K)+C(I,J,K)
10   CONTINUE
```

In this instance the programmer wishes to operate on the entire matrices A, B, and C. It is obvious that the order of the subscripts is not important for

```
     A(J,K,I)=B(J,K,I)+C(J,K,I)
```

will yield the same results as the previous assignment statement, provided the matrices A, B, and C are not equivalenced or overlapped in some ridiculous COMMON block allocation.

The simplest subarray representation for this case would be

```
     DIMENSION A(100,100,100),C(100,100,100),B(100,100,100)
     .
     A(*,*,*)=B(*,*,*)+C(*,*,*)
```

A detailed specification of forms and uses for subarray notation can be found in chapter 10 of Volume III (FMP FORTRAN REFERENCE MANUAL). The case shown here uses the asterisk (*) to represent the fact that the full subscript range is to be used, hence the ADD operation will take place over the entire array. The compiler and hardware can perform this function according to whatever subscript order is optimal for that architecture!

An array variable may appear without any subscripts, in which case the entire array is processed in normal subscript order. Thus

```
     A = B + C
```

is eqivalent to

```
     A(*,*,*)=B(*,*,*)+C(*,*,*)
```

in the example above.

What if the need were to process only a single column of the matrices?

```
     DIMENSION A(100,100,100),B(100,100,100),C(100,100,100)
     .
     DO 10 I=1,100
10   A(I,1,1)=B(I,1,1)+C(I,1,1)
```

This sequence would perform the single-column addition of B and C. In subarray notation this could be stated

```
     DIMENSION A(100,100,100),B(100,100,100),C(100,100,100)
     .
     A(*,1,1)=B(*,1,1)+C(*,1,1)
```

A significant feature of this notation is that a programmer working on a large code can spot such a statement in the middle of many pages of source code and know instantly that a columnar operation is being invoked, without having to hunt upstream in the listing (perhaps for several pages) to find a related DO loop.

So much for the shorthand programming of DO loops provided by the asterisk operator, which can appear in any or all of the subscript positions in an array reference. What happens if one wishes to deal with portions of the array other than the commonly encountered:

```
A(*,*,1)-----the first vertical plane of the mesh
A(*,1,*)-----the first orthogonal plane of the mesh
A(1,*,*)-----the first horizontal plane of the mesh
```

What if only the interior points in the matrices were to be processed leaving all of the outside planes untouched? In normal FORTRAN this would become

```
     DIMENSION A(100,100,100),B(100,100,100),C(100,100,100)

     DO 10 I=2,99
     DO 10 J=2,99
     DO 10 K=2,99

10   A(I,J,K)=B(I,J,K)+C(I,J,K)
```

Using subarray notation the three-level, nested DO loop could be replaced by

```
A(2:99,2:99,2:99)=B(2:99,2:99,2:99)+C(2:99,2:99,2:99)
```

where the normal subscript is replaced by the construct

```
n1:n2
```

n1=starting subscript value (may be integer constant or variable expression), n1≤n2

n2=ending subscript value (integer constant or variable expression), n1≤n2

If every other element in the interior mesh were to be skipped, the optional construct

```
A(2:99:2,2:99:2,2:99:2)=B(2:99:2,2:99:2,2:99:2)+C(2:99:2,
 2:99:2,2:99:2)
```

is permitted which is represented in general by

```
n1:n2:n3
```

where n3 is the increment value for that subscript, which may be an integer constant or integer variable expression, and must be greater than zero.

Now this latter example certainly doesn't appear to be a shorthand version of the desired processing, yet when DO loops must be inserted in "dusty deck" FORTRAN to assist in vectorization, this construct can be a better alternative. More importantly it forces the programmer to think "parallel" in terms of "planes", "solids", and other forms of the subarray. Further, the pathological subarray case just given doesn't appear very often in real code and thus shouldn't frighten programmers from the use of the construct.

How is this notation used in practice? In the recoding of the implicit code, the original flow variable mesh is treated in "slabs" which can be fit into Main Memory. A slab could be several columns in the L direction of the mesh, with each slab consisting of JMAX times KMAX elements. If static FORTRAN variables were used (and they were not used in the actual implicit recoding), the following might appear in a sample code.

```
     DIMENSION Q(100,100,6,100),X(100,100,10)
     .
     DO 10 L=1,LMAX,10
     X(*,*,*)=Q(*,*,1,L:L+9)
     .
     .
10   CONTINUE
```

In this example, every pass through the loop another slab of 10 columns is moved to the temporary matrix X. This case was given to show the admixture permitted of fixed subscripts (the ,,1, in the Q reference), asterisks, and subarray subscripting with integer variables (L:L+9).

The subarray notation is a powerful tool for assisting the compiler and for documentation purposes. It was used extensively in the implicit code vectorization to demonstrate its use in a variety of cases. Subarray notation is also key to the implicit and explicit definition of DYNAMIC variables, to be discussed in a later section of this language description.

3.2.2 THE ANSI 77 SPECIFICATION

Review of the reference manual in Volume III will show that each ANSI 77 revision appears on a page separate from the original base document, with letters keying the location of each insertion into the reference manual. The significant changes of the ANSI 77 effort worth noting are:

- Major revisions in the I/O forms, and use of internal files to eliminate ENCODE/DECODE statements.
- The "Zero-Trip" DO loop, which tests the DO variable at the beginning of the loop instead of the end.
- The addition of TYPE CHARACTER.

3.2.3 THE CYBER 200 FORTRAN 77 ADDITIONS

Incorporated in the revision pages, along with the ANSI 77 insertions, are several additions felt necessary to fully support the CYBER 200 computer family, and also the FMP. The most significant item in this category is the inclusion of a HALF PRECISION variable and array type plus half-precision implicit functions. Both the CYBER 200 and the FMP can utilize the 32-bit and 64-bit formats available for faster arithmetic throughput, as well as more compact storage for some arrays. The addition of HALF to FORTRAN 77 is essential to providing access to this feature.

3.2.4 FMP FORTRAN EXTENSIONS

Although the base document revised as stated above would offer a fairly rich programming language, it was found that a few additional extensions designed primarily for the FMP would improve the chances that compiled code would make optimum use of the FMP. These extensions consist of two declaratives, LEVEL and DYNAMIC, and one executable statement, DEFINE. Although these extensions are described in the FORTRAN reference manual (Volume III), they are crucial to the recoding performed on the three-dimensional implicit and explicit flow codes supplied by Ames. Some tutorial commentary thus seems necessary at this point to explain why the extensions have been created and how they were intended for use.

3.2.4.1 THE LEVEL STATEMENT

Rationale:

In any computer possessing a hierarchical memory system, where a performance differential exists in the use of each level of the hierarchy, the programmer is faced with the need to make judicious choices in the area of data allocation. It is true that compilers can attempt to automatically allocate data to the hierarchy, and in some virtual memory systems the hierarchy is managed by the operating system, however, a minor error in judgment as to where data should be placed (or which data to be paged) can have major impact on the system performance. For optimal use of the hardware resources, the programmer who knows the actual data flow must allocate and schedule the major data blocks in his program.

The use of the two-level memory in the CDC 7600/CYBER 176 has required that programmers deal with split data allocation in an explicit manner. Note that even with this facility, the compiler and operating system still make use of the second level memory for I/O buffers and subprogram "roll-in", quite independent of the programmer's actions. The concern is for an extension which forces the programmer to consider the data allocation issue directly and then assist the compiler by instructing it where data should be placed. This is even more true in the FMP where the programmer is confronted with three levels of memory and problems that will consume almost all available memory space at each level. Hence, the inclusion of the LEVEL statement.

Form:

A description of the LEVEL statement can be found in chapter 6 (page 6-3.1A) of Volume III (FMP FORTRAN specification). The LEVEL statement is a declarative, used to assign variables and arrays to a specific level of memory. The default allocation of data, in the absence of a LEVEL statement is always to Main Memory (level 1).

```
LEVEL n Vnam1,Vnam2,...,Vnamm
```

where n is an integer or integer PARAMETER whose values can be

1-----Main Memory
2-----Intermediate Memory
3-----Backing Store

and Vnam1,...,Vnamm are symbolic names of variables, arrays, dynamic variables, or dynamic arrays (dynamic types will be discussed in a subsequent section).

Scalar variables may be allocated to either level 1 or level 2 memory but not to level 3 (Backing Store). This is because the Backing Store is only accessed in large blocks, and a single scalar reference would require moving an entire block to or from the Backing Store, creating a very inefficient use of that system.

Examples:

```
LEVEL 1 A,B,C(100,100)
```

This statement would assign the scalar variables A and B to Main Memory and the 10,000-element array C to Main Memory.

```
DIMENSION X(100,100),Y(100,100)

LEVEL 3 X,Y,Z(100,100)
```

This level statement would assign the three arrays X, Y, and Z to the Backing Store.

```
LEVEL P1 X,Y,Z
```

This statement would assign the variables X, Y, and Z to either LEVEL 1 or level 2 memory depending on the value of the integer parameter P1 (see PARAMETER statement in Volume III). If the value of P1 is greater than 2 at compile time, a compiler diagnostic will be produced and the data will be assigned to LEVEL 1 memory by default.

3.2.4.2 DYNAMIC VARIABLES

Rationale:

Two major stumbling blocks are encountered in an attempt to convert existing algorithms and programs to a vector machine of the CYBER 200 type. First, since it is most efficient to process whole meshes (or at the very least major subarrays of such meshes) to maximize utilization of the many parallel elements in today's machines, the language must provide facilities for dealing in the largest subarrays practicable. The fixed DIMENSION statement makes it difficult to move subarrays about while ensuring maximum contiguous storage of data. Second, the conversion of simple scalar variables to array (or 'slice') form requires converting all scalar references to array references. These two items need to be dealt with for an effective FMP implementation.

First difficulty, first: memory space is wasted when problem variable dimensions are less than the maximum specified by static variable DIMENSION statements.

Given the statements

```
     DIMENSION Q(100,100,6,100),X(100,100,100),Y(100,100,100),
   1   Z(100,100,100)
     .
     .
      KMAX=50
      JMAX=50
     LMAX=50
     .
     DO 10 J=1,JMAX
     DO 10 K=1,KMAX
     DO 10 L=1,KMAX
     .
     READ(TAPE 1)(Q(J,K,1,L),X(J,K,L),Y(J,K,L),Z(J,K,L))
     .
     .
10   CONTINUE
```

the data stored into each of the arrays Q, X, Y, AND Z will not be in contiguous locations. Arrays X, Y, and Z will each have data stored in the first 50 elements (a half-column) of the first 50 columns of the first 50 planes. But, since the dimensions are 100 x 100 x 100, each of the first 50 planes will have half-columns of data followed by half-columns of empty space for the first 50 columns, and this will be followed by a half-plane (5000 elements) of empty space before the data for the next plane begins. In addition, the last 50 planes will be totally empty (500,000 elements) making a total of 875,000 (50 x 50 x 50 + 100 x 50 x 50 + 100 x 100 x 50) empty elements discontiguously through each of the X, Y, and Z arrays (1,000,000 elements each). In a similar manner (but more complex because of a fourth dimension), the Q array will have a total of 5,875,000 empty memory elements through the total of 6,000,000. Again, these will be discontiguous in sizes from 50 to 3,055,050 elements.

This leads to a waste memory space being unused because of the 'static' definition imposed by the DIMENSION statement. In addition, the longest vector possible in this case (without performing a gather operation) would be 50 elements. If the data could be stored contiguously as though the DIMENSION statement in this instance were

```
     DIMENSION Q(50,50,6,50)
```

subarray operations of the form

```
     Q(*,*,1,1)=Q(*,*,1,1)**2
```

would invoke a single vector operation of length 50*50 elements.

Most FORTRAN compilers provide for variable dimensioning in subprograms as:

```
     DIMENSION A(L,M,6,N)
```

where L, M, and N are normally passed as parameters. However, the dimensions normally cannot be changed during execution of the subroutine. A preferred method would be the ability to 'reshape' arrays dynamically during any subprogram execution to maximize the use of the FMP Vector Units, and to improve the data storage demands.

The second difficulty revolves around the desirability to transform scalar algorithms as directly as possible, with little intervention. For example, the original code includes:

```
     DIMENSION A(100,100,100),B(100,100,100)
     .
     DO 10 J=1,100
     DO 10 K=1,100
     DO 10 L=1,100
     .
     A11=A(J,K,L)
     A12=A(J,K-1,L)
     A13=A(J,K+1,L)
     A(J,K,L)=A11*(A12+A13)
10   CONTINUE
```

This loop could be vectorized in J but not in the K direction because of the recursion there. For simplicity's sake it would be desirable to let the original scalar variables A11, A12, and A13 become vector variables of length 100, without the need to insert a new dimension statement as might be normally required:

```
     DIMENSION A(100,100,100),B(100,100,100)
     DIMENSION A11(100),A12(100),A13(100)
     .
     DO 10 K=1,100
     DO 10 L=1,100
     .
     A11(*)=A(*,K,L)
     A12(*)=A(*,K-1,L)
     A13(*)=A(*,K+1,L)

     A(*,K,L)=A11(*)*(A12(*)+A13(*)
10   CONTINUE
```

To alleviate the difficulty involved in a vectorization effort, a new data type was created--DYNAMIC--which represents arrays and subarrays whose dimensions are established at execution time instead of at compile time. The first usage is shown by restating the previous example:

```
     DIMENSION A(100,100,100),B(100,100,100)
     DYNAMIC A11,A12,A13
     .
     DO 10 K=1,100
     DO 10 L=1,100
     .
     A11=A(*,K,L)
     A12=A(*,K-1,L)
     A13=A(*,K+1,L)
     A(*,K,L)=A11*(A12+A13)
10   CONTINUE
```

Here the original scalar variables have been declared DYNAMIC, meaning that they become pointers to subarrays which will be allocated at execution time in the area of memory called 'DYNAMIC SPACE'. This memory area is what remains in each hierarchical level memory after all code, scalar variables, statically dimensioned arrays, buffers, and sundry system data are allocated.

The beginning of current dynamic memory is pointed at by a canonical register in the FMP register file called the Dynamic Space Pointer. As data space is needed for temporary variables and vectors by the object code, space is allocated and the pointer updated. In this example 100 words of dynamic space would be allocated for each of the dynamic variables A11, A12, and A13. The dynamic space pointer would be updated to point at the next free space, and the data movement from the array A to each of the respective 'slices' would be initiated.

The variables A11, A12, and A13 would be assigned a 'shape' with a single dimension of length 100. (A dynamic variable may have up to seven dimensions ascribed to it to represent the 'shape' of the data area in dynamic space being pointed at.) Figure 18 gives a representation of the memory allocation of the pointers which are the DYNAMIC variables, and the data area being pointed at by the DYNAMIC variables.

Note that the shape of each of the DYNAMIC variables A11, A12, and A13 is established implicitly by the variable appearing as the object of an arithmetic assignment statement, whose source is a subarray or subarray result. The shape can be changed implicitly as many times as the variable appears as an object of an assignment statement:

```
    DIMENSION A(10,10)
    .
    DYNAMIC X
    .
    X=A(*,1)
    .
    X=A(*,*)
    .
    X=A(2:5,2:5)
    .
```

In the first appearance in this example, the variable X becomes a pointer with a shape of one dimension and a length of 10.

At the next occurrence X becomes 'reshaped' into a two-dimensional data space with dimensions 10 by 10 elements. Finally, the last subarray reference again reshapes X as a two-dimensional data space of length 4 by 4 elements.

C-2

```
DIMENSION  A (10,10)
DYNAMIC    X, Y(3)
     •
     •
     •
X=A(*,1)
Y(2)=A(1,*)
```

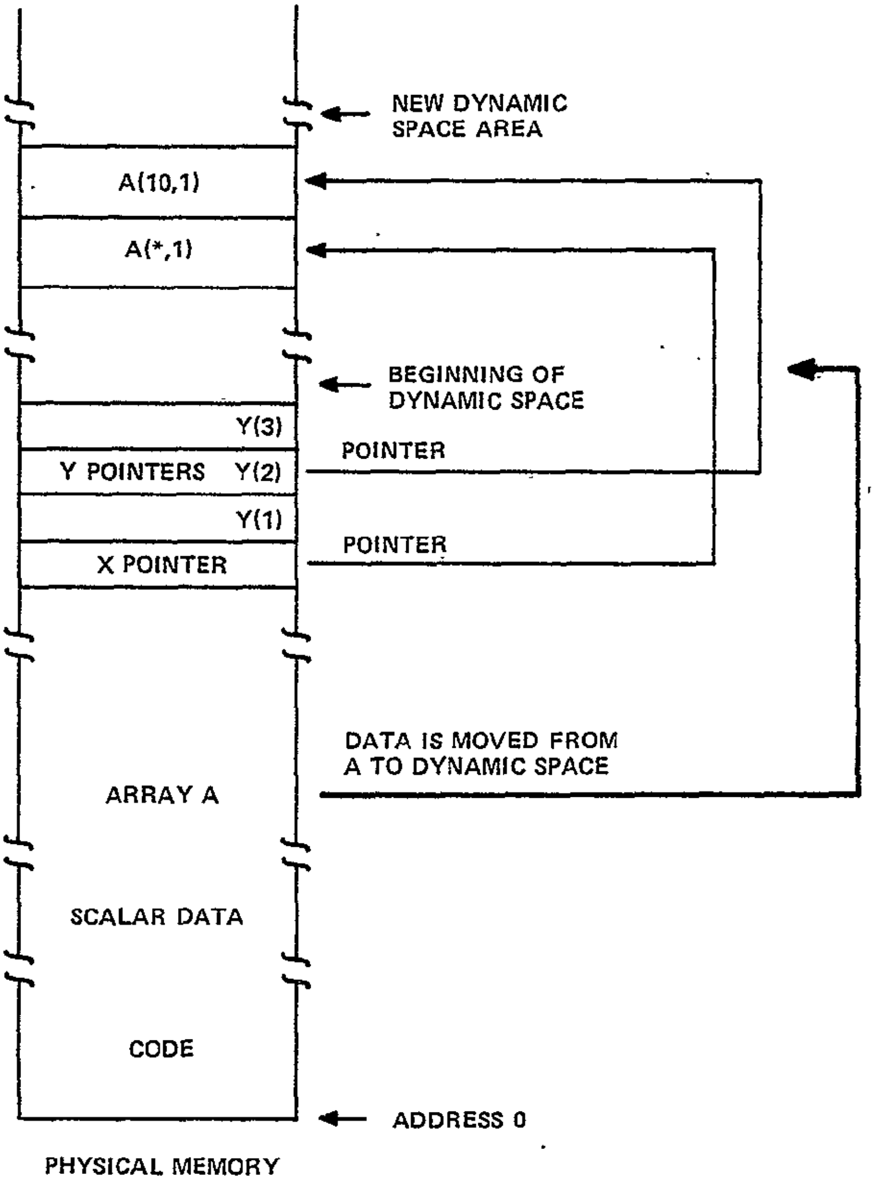


Figure 18. Memory Allocation and Assignment of DYNAMIC Variables

The implicitly defined shape can be 'passed' on to other DYNAMIC variables as in:

```
DIMENSION A(10,10)
DYNAMIC X,Y,Z
.
X=A(*,1)
Y=X*X
Z=Y**2
```

In this example Y and Z take on the same shape as X, and new data space is allocated for each in dynamic space before the calculation is performed.

Conformability:

Implicitly defined DYNAMIC variables as shown so far must obey the rules for conformability when appearing as the source operands for assignment statements, but obviously when they appear as objects of assignment statements they are always reshaped, and thus automatically obey the conformability rules for matrix operations:

```
DIMENSION A(10,10)
DYNAMIC X,Y,Z
.
X=A(*,1)
Y=X*A(*,2)
```

The multiplication of the subarray A by the DYNAMIC variable X is conformable because all dimensions are congruent. The statement:

```
Y=X*A(1,*)
```

would also be conformable since the subarray A(1,*) is a one-dimensional vector of length 10 (albeit requiring a gather operation to form the vector) as is the subarray A(*,1). However, the statement

```
Y=X*A(2:5,2:5)=X
```

is non-conformable because the array reference is static and cannot be reshaped, and does not match the shape of X in dimensions or size. This occurrence would also cause a fatal object-time diagnostic.

3.2.4.3 EXPLICIT DEFINITION OF DYNAMIC VARIABLES

Rationale:

There are instances when the reshaping of DYNAMIC variables should be more controlled than that which is permitted by implicit definitions arising from arithmetic assignment statements with DYNAMIC variables as objects. In addition, many times it is desirable to neither allocate space in dynamic space nor to move data to a working space unnecessarily, despite the fact that it is dynamically structured. To provide more explicit control over DYNAMIC variable definitions the DEFINE statement is provided:

```
DIMENSION A(10,10)
DYNAMIC X,Y,Z
.
DEFINE (X,A(*,1))
.
X=X**2
```

This sequence causes the variable X to become a pointer to the subarray A(*,1) which is actually in place in the array A, rather than in dynamic space. The DEFINE statement is an executable statement which may appear any place in a FORTRAN program where any other executable statement may appear. Upon execution, it explicitly shapes the object dynamic variable X and assigns the address of the subarray within A. Figure 19 shows a representation of this definition in physical memory. No data motion takes place as a result of the DEFINE statement.

The subsequent arithmetic statement X=X**2 then performs the squaring of the subarray A(*,1) and replaces the original subarray with the result, no dynamic space being allocated. The statement

```
X=A(*,3)
```

would replace the subarray A(*,1) with the subarray A(*,3).

```
DIMENSION A (10,10)
DYNAMIC    X,Y(3)
     •
     •
     •
DEFINE (X, A(*,1))
DEFINE (Y(2),A(10,1))
```

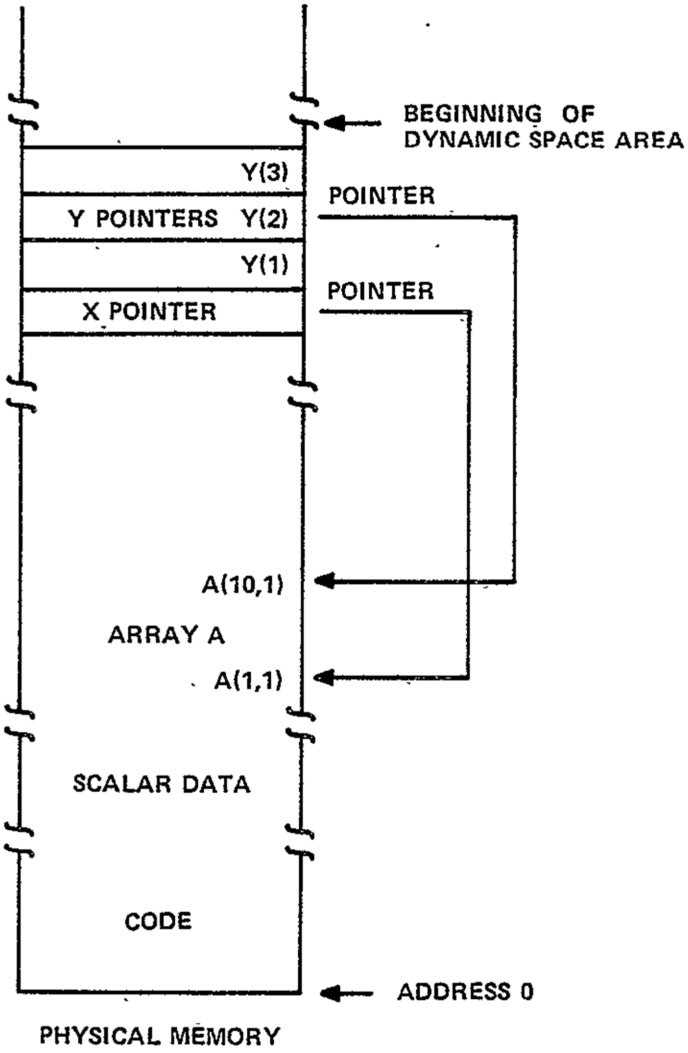


Figure 19. Alternative Memory Allocation Using DEFINE Statement

Conformability:

Explicitly defined DYNAMIC variables can only be reshaped by execution of another DEFINE statement, and not by appearing as the object of an assignment statement. Thus such explicitly defined variables must obey the conformability rules on both sides of the equal sign in an assignment statement:

```
     DIMENSION A(10,10)
     DYNAMIC X,Y,Z
     .
     DEFINE (X,A(*,1))
     .
     Y=A(*,2)
     .
     X=A(*,3)
     .
     Y=A(2:5,2:5)
     .
     X=A(2:5,2:5)
```

The reshaping of Y is permitted because it is implicity defined; however, the reshaping of X=A(2:5,2:5) is not permitted since its shape has been fixed by one explicit DEFINE. The execution of another DEFINE statement:

```
     DEFINE (X,A(2:5,2:5)
```

would change the shape (and size) legally.

Forms:

The variety of permitted statement forms for DYNAMIC and DEFINE are detailed in the FORTRAN specification (Volume III, page 6-3.2A and page 10-3.1A, respectively). Basically the forms are

```
     DYNAMIC Vnam1,Vnam2,...,Vnamn
```

where Vnam1...Vnamn represent a list of variable or array names which are to be established as dynamic pointers.

```
     DEFINE (DVnam,subarray reference),(DVnam...........

     DEFINE (DVnam,DVnam),(DVnam,.......

     DEFINE (DVnam,(subarray subscripts)),(.......

     DEFINE (DVnam,DAnam(subscripts))
```

where DVnam is any DYNAMIC VARIABLE name and DAnam is any DYNAMIC ARRAY name.

The first DEFINE form has already been illustrated. Note that the objects (left-hand member of each pair) must be a DYNAMIC variable or an element of a DYNAMIC ARRAY.

The second example is used to make one DYNAMIC variable synonymous with another. This means <u>synonymous</u> but not <u>identical</u>:

```
DIMENSION A(10,10)
DYNAMIC X,Y,Z
.
.
DEFINE (X,A(*,1))
.
DEFINE (Y,X)
.
DEFINE (X,A(*,4))
.
```

In this example X is first defined as pointing to the first column of A. Y is then made synonymous to X, which means the pointers for Y are set the same as those for X. However, the last statement redefines X while the definition for Y will still point to A(*,1).

The third form of the DEFINE is used to 'fix' a dynamic allocation of data and definition of the DYNAMIC variable's 'shape'.

```
DEFINE (DVnam,(subarray subscripts),(....
```

In this case the DVnam may be the name of a DYNAMIC variable of a DYNAMIC ARRAY element (see next section for DYNAMIC ARRAYS). The subarray reference may include up to seven subscript expressions, any one of which may be a subarray form:

```
DYNAMIC A,B
.
DEFINE (A,(1:10,1:10,1:10))
.
DEFINE (B, I:J,1:100,I:100))
.
```

The first DEFINE statement causes an allocation of 1000 elements of dynamic space to the variable A, with the shape of 10*10*10 elements. No data movement takes place. In this regard, DEFINE performs the same function as an 'implicit' definition but without the data movement. The resulting memory allocation and pointer setup is the same as shown in figure 18, for implicit definition. The major difference here is that the variable A cannot be reshaped by appearing in an assignment statement. The statement

```
A= X(1:5,5:10,10:100)
```

would therefore result in an object-time fatal diagnostic message.

The second DEFINE statement results in a similar allocation of dynamic memory and the shaping of the variable B, but the amount of space allocated and the shape will not be known until the object-time execution of the DEFINE is accomplished and the values of the variables I and J are known.

The remaining forms will be discussed with the final construct of these extensions--DYNAMIC ARRAYS.

3.2.4.4 DYNAMIC ARRAYS

Rationale:

As seen from the previous discussion and by a glance at the recoding of BTRI in the implicit code, much scalar code can be readily transformed into vector code while preserving its original appearance. There are cases that cause some amount of distress, however. Take the example:

```
     DIMENSION A(100,100),B(100,100),F(5)
     .
     .
     DO 10 I=1,5
     DO 10 J=1,100
     DO 10 L=1,100
10   F(I)=A(J,L)+B(J,L)
```

If the loop is to be vectorized in both J and L, the variable F must be redefined as a series of 5 vectors, each with length 10000.. To define F as a DYNAMIC variable, five occurrences of the variable F, one through five must be provided. This creates the concept of an array of DYNAMIC variable pointers called a DYNAMIC ARRAY. This is accomplished by permitting the DYNAMIC statement forms

```
DYNAMIC A,B,F(5)
```

or

```
DIMENSION F(5)
DYNAMIC A,B,F
```

This will create an array space for pointers which may be addressed by subscripting the variable name F. An implicit definition of a DYNAMIC array element can thus be done as

```
DIMENSION A(10,10,10),B(10,10,10)
DYNAMIC F(5)
.
F(3)=A(*,*,*)
.
```

The third element in the dynamic array F will be allocated dynamic space and given the same shape as the static array A - 10*10*10 elements. Note that each of the pointer elements in F need not be related, spatially or logically, to any other element. Thus additional statements

```
.
F(1)=B(*,1,*)
.
F(2)=A(1:5,5:10,5:10)
.
```

would set up other pointers in F to allocated space in dynamic memory that may not be contiguous--that is the physical memory allocated for F(1) may not be contiguous with the physical memory allocated for F(2), since the allocation is done dynamically as the respective assignment statements are encountered during execution of the program.

3.2.4.5 EXPLICIT DEFINITION OF DYNAMIC ARRAY ELEMENTS

The forms of DEFINE statement given in the preceding section include the meta-symbol 'DVnam' representing a DYNAMIC variable name. In the preceding examples this was limited to simple DYNAMIC variables. It should be obvious that any DYNAMIC array element can be used in place of such simple DYNAMIC variables:

```
DYNAMIC A,B(10)
.
DEFINE (B(3),(1:10,1:10,1:10))
.
DEFINE (A,B(3))
.
DEFINE (B(I),(J:K,J:K,1:5))
.
```

In these examples the DYNAMIC array elements B(n) can be used interchangeably with scalar DYNAMIC variables in the first three DEFINE forms shown.

The last (fourth) DEFINE form

```
DEFINE (DVnam,DAnam(subscripts))
```

has a special function which requires that DVnam must be a simple DYNAMIC variable. This is caused by the fact that many times a subarray reference is desired for a dynamically assigned variable:

```
DIMENSION A(10,10,10)
DYNAMIC X,Y
.
DEFINE (X,A(*,1,1))
.
Y=X(1:5)
.
```

In this case, access is wanted to a subarray of the dynamically defined variable X (which is itself a subarray of the static array A). If however, access is desired to a subarray of a DYNAMIC array element, the constructs would have to look like

```
Y=Z(3)(1:5)
```

where the third element of the DYNAMIC array Z is used instead of the simple DYNAMIC variable X. This construct is considered messy to read, and makes FORTRAN scanning and error detection quite difficult in the general case. Therefore the methodology for getting at subarrays of DYNAMIC array elements requires 'aliasing' the DYNAMIC array element to a simple DYNAMIC variable and then using the variable:

```
DIMENSION A(10,10,10)
DYNAMIC X,Y,Z(5)
.
DEFINE (Z(3),A(*,1,1))
.
DEFINE (X,Z(3))
.
Y=X(1:5)
.
```

In this case the variable X was made synonymous with the shape and location of the third element of the DYNAMIC array Z. This 'aliased' variable X is then used as the basis for the subarray reference in the assignment statement which moves data to Y.

This methodology makes some references somewhat cumbersome at first sight, but the usage is normally limited to several instances in a program and doesn't appear to create a great burden on the programmer.

An important sidelight to the use of DYNAMIC arrays was mentioned previously, that being the fact the no single DYNAMIC array element need be related to any other. Using DEFINE statements one can establish particular relationships between an entire DYNAMIC array and the object that its independent elements are describing. For example

```
DIMENSION A(100,100,100)
DYNAMIC F(6),B
.
DEFINE (F(1),A(1,*,*))
DEFINE (F(2),A(100,*,*))
DEFINE (F(3),A(*,1,*))
DEFINE (F(4),A(*,100,*))
DEFINE (F(5),A(*,*,1))
DEFINE (F(6),A(*,*,100))
.
.
B=F(4)*F(5)
```

This example assigns each of the exterior planes of the mesh to a different F pointer. The arithmetic expression then performs operations on the planes as shown. An interesting side effect of this is a very powerful statement:

```
F=F**2
```

wherein all subarrays in F are processed with a single arithmetic statement.

3.2.4.6 SUBROUTINE COMMUNICATION OF DYNAMIC VARIABLES

The processing of DYNAMIC variables and DYNAMIC array elements proceeds interpretively at execution time, using pointer information in a set of 16 words allocated to each variable or element. Fourteen words contain the shape of each of 7 dimensions (in subarray notation, the starting index, ending index, and increment); the remaining two words contain a count of the number of active dimensions, base address of beginning of subarray, and the maximum space allocated for this variable's data in dynamic space. Thus, when a DYNAMIC variable appears in COMMON or as a parameter, 16 words are set aside for each DYNAMIC element. Obviously, such elements must be defined as DYNAMIC in all routines using them, and in the case of COMMON block transmission of data, any routine having a block COMMON containing a DYNAMIC variable in it must define the variable as DYNAMIC.

DYNAMIC variable names may not appear in EQUIVALENCE statements. When such variables appear in INPUT/OUTPUT statements they point at the data to be transmitted, the pointers themselves are never output or input, their life is limited to the dynamic execution environment of the operating program. Debugging tools permit the programmer to examine and modify the shape described by the DYNAMIC variables, however.

3.2.5 COMPILER STRATEGY

The FMP FORTRAN compiler will consist of all facilities already expected for FORTRAN compilers for large scale computers in the 1970s for the validation and evaluation of source language statements, generation and scheduling of object code, and production of diagnostics and debugging aids for the user's programs. In addition, the compiler must be able to accept and evaluate the several FMP language extensions discussed above. The most crucial objectives for the compiler revolve around its ability to optimize the parallel execution of the Map and Vector Unit instructions. Illustrations of the implicit code in this report have assumed that the compiler would be able to generate the appropriate object code to achieve the maximum overlap. It has been further stated that the compiler could not be expected to automatically recognize the optimum "slabbing" and create object code to perform the "slabbing" functions without the assistance of the programmer. To that end the language extensions were created to make the hierarchical memory and data structures visible to the programmer and to make the programmer responsible for the management of data mapping.

In exchange for this, the compiler is expected to organize and schedule the code and to maximize machine performance. The first part of the recoded STEP routine (appendix D) will now be examined to see what the compiler must do in code generation and to estimate the final code performance. In section 5 the actual simulation of this code will be covered.

3.2.5.1 DO LOOP "GET READY"

Most modern compilers generating code for multi-register machines are capable of generating "prefetch" (load from memory) instructions which are extracted from the DO loop. These instructions then are scheduled to be issued <u>before</u> entering the DO loop. Then at the top of the actual executing DO loop another set of fetch operations are created whose intent is to load the data for the <u>next</u> pass through the loop. This technique then reduces the wait time required while scalar data is being transferred from memory to the high speed registers.

The counterpart of this method for the FMP is to generate the map instructions for the first pass through the loop, then at the beginning of the actual loop to place a set of map operations for data to be used on <u>the next pass</u> through the loop. For example, lines 930 through 960 in appendix D are map operations which become gather record functions with record lengths of LSL*KMAX elements. JMAX records are gathered for each map operation to form the vectors that are processed in the balance of the metric computation (in-line XXM). The compiler will generate these four map operations with the destination data going into Main Memory at the locations designated by descriptors called, respectively, RJ, XKL, YKL, and ZKL. These instructions will be scheduled to be executed ahead of the executable DO loop that begins at statement 830.

In addition, another set of map operations will be generated and scheduled after statement 830. These map operations will perform the data transfers from LEVEL 2 storage for the next pass through the DO loop. The destination areas for the data from these map operations will be designated by a set of auxiliary pointers, invisible to the programmer which could be called RJ', XKL', YKL', and XKL'. At the end of a particular pass through the loop, these pointers are exchanged for those which point to data areas RJ, XKL, YKL, and XKL. In this way, at the expense of a brief exhange of pointer data, the map operations can be overlapped.

Note that a total of 14 slabs of data have to be moved from Intermediate Memory to Main Memory, but there is no data interdependence among these various slabs. Thus each map operation would carry a different dependency key and thus each can be issued immediately to the Map Unit, up to the extent of the queueing buffer in that unit. Assume, for example, that the map operation at line 940

```
XKL = X(*,L-1:L+LSL+1,*)
```

is generated with a dependency key of "01". Instructions continue to be issued after this map operation until the first arithmetic instruction in the loop (line 970) is encountered. Since this instruction references array XKL it will also have a dependency key of "01" imbedded in it. This instruction would then be held up until the map operation with that key is complete. The next arithmetic operation would then be released to the Vector Processor and the process continued. In this case the map operation at line 950

```
YKL = Y(*,L-1:L+LSL+1,*)
```

might carry a dependency key of "02". Then the vector operation using YKL (line 950) would be held up until the corresponding map operation is complete. All of these dependencies would exist only during the first pass of the loop, since on subsequent passes the data would already have been mapped in by the overlapped map instructions to RJ', XKL', YKL' ....etc.

With this form of code generation one can see that not only are almost all map operations overlapped, but due to the action of the compiler and the dependency keys, a good deal of the prefetch map operations have some of their execution overlapped, so that loop startup is minimized. For example, consider the previous sequence. Before the first arithmetic operation can begin, the entire slab for XKL must be mapped into Main Memory. This gather record operation will require JMAX*KMAX*(LSL+2)*3/8 clock cycles to complete. Once this data has been moved the next map operation can begin for the YKL slab. Meanwhile the subtract and multiply operation

```
XK = (XKL(3:KMAX+2,2:LSL+1,*)-XKL(1:KMAX,2:LSL+1,*))*DY2
```

will be initiated. The time required for this instruction would be

```
JMAX*KMAX*LSL/8 clock cycles.
```

This is approximately 3 times faster than the concurrent map operation, thus the next arithmetic function:

```
YK = (YKL(3:KMAX+2,2:LSL+1,*)-YKL(1:KMAX,2:LSL+1,*))DY2
```

must wait the completion of the corresponding map operation. This wait for the dependency key to become free continues for statement 990 as well. Thus on the second through the last time through the loop, the execution of statements 970 to 990 requires

```
3*JMAX*KMAX*LSL/8 cycles
```

but on the first pass (due to the wait for data from LEVEL 2 memory) these same statements require

```
3*JMAX*KMAX*(LSL+2)*3/8+JMAX*KMAX*LSL/8 clock cycles.
```

The statements at 1000 through 1020 also reference the already mapped arrays XKL, YKL, and ZKL and thus proceed at maximum rate. Meanwhile the map operation

```
RJ = Q(2:KMAX-1,L:LSL,6,*)
```

would be issued with a dependency key of, say, "04", and be fully overlapped with the operations at 1000 through 1020. If everything issues without hidden conflicts, the statements at 1030 through 1050 should be executed without waiting since the data RJ should be completely mapped into Main Memory by the time the Vector Processor is ready to execute statement 130. If an approximation of three vector instructions for each gather record (map) instruction is used, assuming equal vector lengths, it can then be seen that the map instruction for

```
RR = 1./Q(2:KMAX-1,L:L+LSL,1,*)
```

will have been completed by the time the Vector Processor is ready to proceed with the divide operation. The same is true for the other map operations used to gather data for arithmetic in statements 1150 through 1210. It is possible then to compute the prefetch overhead for the first pass through the J sweep and L sweep loops. This becomes

```
3*JMAX*KMAX*(LSL+2)*3/8-(2*JMAX*KMAX*LSL)/8 =
  (7*LSL+18)*JMAX*KMAX/8 clock cycles, loop overhead.
```

The important thing to note here is that this overhead only occurs during the first trip through the first and third loops, and constitutes the only visible cost of performing map operations on the FMP; it results from the compiler scheduling map instructions so that the set of map data required for the next trip is available before being required. (The overhead for loop 2 is discussed in the next paragraph.) Thus in the example, where JMAX = KMAX = LMAX is used, the number of slabs (or trips through the loops) is 17. The overhead shown is therefore amortized over all of these trips through the loop to complete the sweep.

For the K sweep direction, the single-element gather operation dominates the loop overhead to a much greater degree; all of the operations for statements 2090 through 2330 are completely constrained by their corresponding map operations. In this case the compiler will generate 23 arithmetic operations plus one divide, with the divide requiring the same number of cycles as two arithmetic operations. Thus the arithmetic time would be approximately

25*KMAX*LMAX*JSL clock cycles

while the Map Unit, in performing the corresponding 9 gather element operations, would take

9*KMAX*LMAX*(JSL+2)*6 clock cycles.

The overhead for the first pass would then be

54*KMAX*LMAX*(JSL+2)-25*KMAX*LMAX*JSL =
KMAX*LMAX*(29*JSL+108) clock cycles.

Total overhead for all three sweep directions then becomes

OVERHEAD = KMAX*LMAX*(29*JSL+108) + JMAX*KMAX*(7*LSL+18)/8
+ JMAX*LMAX*(7*KSL+18)/8 clock cycles.

3.2.5.2 INFERRED TRANSPOSE OF MATRIX

Aside from the judicious scheduling of map and vector instructions, the compiler has one additional burden placed upon it. The compiler must be able to evaluate loop 8 (statements 1900 through 2020) and generate the implied matrix transpose operations. The design of the Map Unit permits a single map instruction to perform the necessary transpose process. The compiler must be able to discern that construct from loops of the type shown in loop 8. The alternative is to add an extension which calls for the explicit transpose, but is unnecessary in this case, since the loop describes the actions desired.

With these simple attributes the compiler should produce code that permits the FMP to achieve its maximum rate.

3.3 COMPILER FUNCTIONAL CHARACTERISTICS

Volume III contains a preliminary document describing the FMP FORTRAN language and its usage. In an attempt to match certain language features to the FMP architecture, the parallel nature of the processing was made visible to the programmer through the DYNAMIC and DEFINE statements and subarray references. The final responsibility for matching the problem statement to the FMP hardware rests with the language processor (compiler), however. To achieve the NASF effectivity goals, the characteristics of the compiler will have to be specified in detail by NASA. The following are suggested items that should be included for elaboration in any NASF compiler specification, or used in consideration of any compiler development proposal.

3.3.1 SOURCE CODE

The FMP compiler must be able to accept input, audit, and produce object code or diagnostics for the complete FMP FORTRAN language as described in Volume III. This language is based on ANSI FORTRAN 77, with extensions that Control Data feels should appear in standard compiler products, such as "hexadecimal" data types, plus extensions felt necessary for the FMP, such as subarray notation.

The compiler should provide a mode wherein all statements not conforming exactly to ANSI 77 standards will cause a warning message to be printed. In addition, the compiler may also provide a mode of operation wherein other extensions available in compilers operating at Ames (from DEC, IBM, and CDC) might be tolerated to provide internal programming compatibility at Ames. The extent of these latter augmentations is unknown at this time and they do not appear in the language description currently called FMP FORTRAN.

The 32/64-bit nature of the Control Data FMP is accommodated in two ways: through use of the HALF PRECISION data type and related FORTRAN supplied functions, and through the use of a compile-time option which can compile an entire program with full precision (normal mode) being 64-bit or 32-bit. In this latter case (32-bit=full precision) HALF PRECISION is considered 32-bit also (since there is no 16-bit floating-point format in the FMP).

The compiler must provide some form of "escape mechanism" to allow the programmer to invoke specific machine-language instructions (except for monitor mode instructions) when the occasion warrants. Although normal applications programs should not have to use such a facility, it is certain that some general applications modules will be "fine-tuned" by clever programmers and will need access to explicit instruction control. There are two means for this:

a) Standard subroutine or function CALL statements to machine-language subroutines--this is the conventional means for handling this problem, however, it implies access to a machine-language assembler by many programmers, and can cause havoc in a large system environment. A second drawback is the execution time cost involved in subroutine CALL sequences. If a programmer desires to invoke but a single peculiar machine function in his code, the encumbrances of writing two separate modules and taking on the subroutine switching overhead at "object time" may be excessive.

b) Imbedded, one-line machine-language statements in the FORTRAN source program, where the instruction uses the variable names and statement labels assigned by the programmer in his FORTRAN code--this approach was used in the STAR FORTRAN language employing the special call notation Q8xxxx where xxxx was a predefined machine-language instruction mnemonic. The parameters of the CALL were in fact the symbolic entries representing each field in an instruction. The machine-language instruction thus specified was then inserted directly into the FORTRAN object code at the point where it was invoked, without the need for an object-time call and return sequence to a separate subroutine. Although this has become a powerful tool in STAR-100 programming, there are several drawbacks:

   - Possible misuse of machine resources--a programmer can unwittingly deadlock the FMP if allowed complete control over such instruction fields as the read and write dependency keys. By permitting access to register file oriented instructions, the programmer may accidentally overflow the available registers. Both of these difficulties can have disastrous consequences.

   - Impeding the compiler ability to generate optimum object code--injection of an "alien" instruction into a FORTRAN sequence may make it impossible for the compiler to automatically vectorize a particular code sequence. In other places such in-line invocation may disrupt the entire instruction scheduling process for scalar, map and vector operations.

c) A third, and the recommended, alternative is to provide a special call syntax similar to the Q8xxxx described above, but limit its application to invoking one of a set of predefined "special functions", which the user can specify at will, but which are imbedded in tables in the compiler, and totally under the compiler control. An example might be

```
PROGRAM DEMO

DIMENSION A(100,100,100)

DYNAMIC B

,

B=Q8XPOS(A)

,
```

which would perform the transpose of the entire mesh A into the dynamic space B. The significant thing is that the method of implementation of the XPOS is left to the compiler, which might choose to use the map instruction, a series of vector operations, or even a scalar loop, depending on what else the compiler was scheduling in the FMP at that time.

The FMP compiler structure should permit the introduction of new functions of the Q8 type, by means of simple table entries that can be augmented as the user desires. The compiler would then attempt the in-line generation of appropriate code sequences from the table "skeletons".

A list of desired Q8 functions has not been assembled at this time for the FMP, since it has sufficed, so far, to efficiently utilize the standard constructs and FMP extensions. The capability should be built into the compiler, however, along with a well defined procedure for requesting and specifying the desired Q8, FMP intrinsic function.

A most significant aspect of restricting the programmer's access to machine functions in this manner is that all FMP programs could be prototyped on machines other than the FMP, given that scalar sequences are implemented for Q8 calls.

3.3.2 OBJECT CODE

Specification of a compiler's object code by a customer is necessarily two-dimensional: volume and speed. For small-to-medium computer systems the amount of object code generated for a given application can significantly affect the space remaining for data. Even in these times of large, relatively inexpensive memory this is a continuing concern of many users. In the case of the FMP this is no longer true. Considering the amount of available Main Memory the expected object code is of such moderate proportions that this factor is not a consideration. Instead, on the FMP the concern is for:

1) maximization of concurrency,

2) use of memory space for problem data and temporary data.

To these issues a compiler specification might address itself with some of the standard verbage:

- "The compiler will utilize the most advanced techniques for generation and scheduling of object code, including common subexpression analysis, invariant code removal, extended basic block optimization, and global analysis of all program modules submitted to a single compilation."

- "The compiler will minimize the amount of storage required to hold temporary vectors, and will optimize the utilization of the critical Main Memory resource."

- "The compiled object code should optimize throughput by maximizing the utilization of the Vector Units; optimization for other units is secondary."

- "The compiler should attempt to 'automatically vectorize' all DO loop constructs that do not include IF, GOTO, CALL, and function references, regardless of the presence of 'recursion' or non-unity increment values for the DO statement."

- "An option will be provided for the compiler to produce object code which uses entirely scalar sequences in place of either or both the Vector Units and Map Units."

The "God and Motherhood" nature of the preceding statements makes clear their purpose, and quite probable their inclusion, in any compiler requirement, proposal, and specification. What additional characteristics should be highlighted for the FMP FORTRAN compiler, however?

3.3.3 CONSTRUCTS

The compiler must be required to recognize certain source language constructs and from them derive FMP instructions. The simple DO loop construct described above would yield

```
     DO 10 I=1,100

10   A(I)=B(I)+C(I)
```

a simple vector addition of the two arrays B and C in Main Memory, with the results going back to memory. By adding a simple statement

```
     DO 10 I=1,100

     C(I)=3.14159

10   A(I)=B(I)+C(I)
```

the compiler should generate two concurrent operations - one map operation transferring the constant to Main Memory array C and one vector add operation. Adding yet another statement

```
      DO 10 I=1,100

      D(I)=E(I)+F(I)

      C(I)=3.14159

10    A(I)=B(I)+C(I)
```

the compiler should produce again two concurrent instructions - one map operation to broadcast the constant to the array C and a vector operation to perform the pair of vector adds simultaneously. A sequence such as

```
      DO 10 I=1,100

      D1(I)=E(I)*F(I)

      C(I)=3.14159

10    A(I)=B(I)+C(I)*D1(I)
```

would also produce one vector instruction and one map instruction, with the vector instruction producing the result D1 which is stored to memory, then using that result internally to form the result A, all in one pass through the data.

A more complex sequence must also be vectorized:

```
      J=1

      DO 10 I=1,100

      IF(A(I).GT.B(I))GO TO 10

      C(J)=A(I)

      J=J+1

10    CONTINUE
```

This construct and its variants should produce a map instruction which performs a vector compress operation, based on the stated conditions. Taking a key example from lines 1900 through 2020 of the implicit code in appendix B:

```
      DO 8 K=1,KMAX

      RJ(1:LMAX-2,1:JSL,K)=Q(K,2:LMAX-1,6,J:J+JSM)

      '

      '

8     CONTINUE
```

This sequence must produce a single map operation for each dynamic variable RJ, XJL, YJL, ZJL which performs the transpose of the arrays. The transpose is accomplished by a gather operation which, for each K, moves LMAX and JSL columns and rows into a new alignment in memory. (See implicit code writeup, Division 2.)

An example of the various forms of object code generated by an FMP-oriented compiler are given in appendix D. The object code lines are denoted by a comment card of the form:

```
C#       VEC nn     op1:op2:op3     VL=n1; WK=kn, RK=km
```

or

```
C#       MAP nn     op4:mm      NR=m1,RS=m2,ST=m3/m4; WK=km,
                                RK=kn
```

or

```
C#       MAP nn     op5:mm      CVL=m5, VL=n1; WK=km,RK=kn
```

where:

nn	abbreviated symbolic name of destination vector
op1,op2,op3	mnemonic vector operation codes
op4	mnemonic code for gather (GTHR) or scatter (SCTR)
op5	mnemonic code for compress (CMPS)
n1	vector length in elements
km	write key
kn	read key
mm	memory option MM=Main Memory to Main Memory IM=Intermediate Memory to Main Memory MI=Main Memory to Intermediate Memory BI=Backing Store to Intermediate Memory IB=Intermediate Memory to Backing Store

m1	total number of records moved
m2	record size (for gather and scatter)
m3/m4	length of stride in each stride direction
m5	control vector length

The read and write keys may be omitted; a key of 0 (no key) is assumed. If any specification field other than read or write key is omitted, a value of 1 is assumed. Only the number of open fields necessary to specify the required function should be used, e.g., MUL:ADD.

The comment line does not include all of the parameters needed for an actual machine instruction, such as addresses, but the code shown represents enough data to feed the FMP simulators. Where it was desired to show the operation relationships to source code symbols, a pair of brackets "<", ">" is used to surround a brief comment about the data used.

When a vector operation produces two outputs, one on AW1 and the other on AW2, two lines are used as in lines 4370 through 4383 of appendix D:

```
      U13=B1(1,3)*L11
      U14=B1(1,4)*L11

C
C#     VEC U13  MUL.  VL=SSL*SSMAX
C*     $$$ U14  MUL
```

Note that C* indicates a continuation of the previous line, and $$$ indicates a dual vector operation. Only one vector length is used for both operations and must appear on the first line.

The code sequences shown in appendix D do not include any of the scalar code, since the concern was primarily with the vector operation rates for simulation purposes. Analysis shows that all scalar operations can be "buried" under the vector execution umbrella IF THE COMPILER SCHEDULES THE CODE PROPERLY (see section 5).

Only one small instance of the compiler scheduling of vector operations is shown and this is critical to the performance of the implicit code. The execution of each sweep calculation in STEP is bound to the data transfers from Intermediate to Main Memory. The compiler must be able to automatically generate and schedule "look-ahead" or "fetch-ahead" code. For a scalar example that is common

```
      DO 10 I=1,100
      D=A(I)**2+B(I)**2
      E(I)=D*(A(I)-B(I))
   10  CONTINUE
```

most modern compilers for multi-register machines will generate a code sequence that looks like:

```
        FETCH A(1) to register A
        FETCH B(1) to register B
        I=1

LOOP:   FETCH A(I+1) to register A'
        FETCH B(I+2) to register B'
        MULT A*A to T1
        MULT B*B to T2
        ADD  T1+T2 to D
        SUB  A-B to T1
        MULT D*T1 to E
        STORE E to E(I)
        MOVE A' to A
        MOVE B' to B
        ADD I=I+1
        TEST
        GOTO LOOP IF NOT DONE
```

This object code is necessitated by the time required in many machines to bring data from memory to a register. The "prefetching" operation helps overlap the time to get the next data from memory with the calculations on the current data. At the end of the computations the new data are then transmitted between registers (a very fast operation when compared with memory transfers).

In a similar manner vectors can be "premapped" so that arithmetic can be overlapped with the next map operation. This is shown in lines 821 through 829 of appendix D where the first set of vectors is mapped into Main Memory before the loop starting at line 830 is initiated.

In the main, the remaining "pseudo object code" shown is left in place with the corresponding FORTRAN source statements, with small exceptions necessitated by the need to combine some operations into a single Vector Unit function. As an example, the object code for line 1150 is shown after line 1160, at line 1162, and is combined with the functions invoked by line 1160 and shown at line 1163

```
1150  U=RR*Q2
1160  V=RR*Q3
1161  C
1162  C#       VEC U  MUL   VL=(KMAX-2)*LSL*(JMAX-2)
1163  C*       $$$ V  MUL
```

In actual practice the compiler must be able to shuffle the generated code around to assure maximum utilization of the Vector Units.

The compiler must provide an object code listing on request, and some method must be provided to key generated code to the source language statements that generate the code. This is essential because more than one source statement may be combined into a single vector operation and then that instruction rescheduled elsewhere in the instruction stream.

3.3.4 PERFORMANCE

The general statement that a compiler "must produce the most efficient object code possible" is not adequate to meet the needs of the NASF procurement. Some definitive and quantitative measures must be established and specified as minimum object code performance goals. The obvious performance goal is to have the compiler and FMP hardware marriage produce an object code execution speed that can complete specified metrics in a certain amount of time. The compiler, however, must be disengaged from the speed of the hardware if it is to be properly specified as a separate, procurable entity. Other measures that suggest themselves are percentage utilization of Vector Unit capacity, percentage used of available concurrency and percentage vectorization. Each of these have some deficiencies. A compiler can generate a totally vectorized code which is terribly inefficient in use of the hardware. A compiler can also generate unneeded vector arithmetic (failing, for example, to eliminate common subexpressions) which keeps the Vector Units 100% busy to no benefit of the actual problem solution. Finally, in a similar manner, the compiler can generate three inefficient streams of code, one each for the Map, Scalar and Vector Units which provide 100% concurrency.

The best alternative seems to be specifying one or more of the performance metrics (implicit, explicit, spectral weather, and finite difference weather codes) as a measure of the compiler capability. Given a coding of any of these metrics in the FMP FORTRAN dialect specified for the NASF, the object code must execute an entire solution without I/O calls in no greater than 120% of the theoretical execution time for the program. This means that a method must be derived for computing the theoretical time allowed.

If the peak rate of the FMP hardware is 1.5 gigaflops, then one can count all arithmetic computations in a metric and determine a best time for execution for a given set of parameters. Where data dependencies exist in the problem solution (as in the method of characteristics) some canonical value and associated parameters could be chosen for the total arithmetic load. Functions such as SQRT, SIN, COS... would each be assigned an equivalent floating-point operation count. For example, if a time is established in this manner for the subroutine BTRI in

STEP, for three calls and mesh dimensions of 100x100x100, this might produce a theoretical execution time of 5.148 minutes.

This approach on the part of the NASF customer is no different than the method for setting performance goals for standard product compiler improvements. A particular benchmark becomes crucial to a sale and the FORTRAN developers are launched forthwith to achieve some real timing goal for that benchmark. To be useful, meaningful segments of production codes must be used for this measurement. Note that the use of I/0 was excluded, an attempt to decouple the object code performance objectives from the operatng system performance objectives. Thee is some danger in this, since a good deal of execution time is spent in that grey area called FORTRAN object-time I/O routines which are not generated by the compiler nor claimed by the operating system developers.

Instead the costs of I/O interface should be measured separately by creating a heavily I/O-oriented benchmark with all desired forms of I/O-formatted, unformatted and direct-and establishing some measure of performance. This measure should include achieving at least 90% of the total I/O hardware bandwidth available, while reducing the execution rate of a fixed number of map and vector operations by no more than 5% (due either to memory interference or object library inefficiencies, or to object time call sequences).

3.3.5 OBJECT LIBRARY

The object code just discussed is that directly produced by the compiler from input source code. In order for the program to execute however, an array of support software is needed to provide the interface to the operating system, I/O system, exception condition hardware (data flag branch register), and the myriad of intrinsic and external FORTRAN functions such as ALOG, MAX, MIN, and the like. A compiler specification must include these items and should establish some minimum measurable goals for these system components.

Generally, object library routines will be manually fine-tuned using either the Q8xxxx FORTRAN extensions, a system programming language such as PASCAL or (ugh!) assembly language. They should therefore make the best use of the machine resources of any object modules executed in the NASF. For the FMP, four considerations should be taken into account in evaluating object library strategies:

1) The compiler should be able to optionally incorporate any of the FORTRAN supplied routines (see Volume III, FMP FORTRAN) in-line, to permit better overlap and optimization of functional unit usage.

2) When incorporated in subroutine form, a maximum allowable code space should be established for each named routine.

3) A minimum performance level should be established in terms of machine cycles per input argument.

4) A set of standard tests for function correctness should be established and verified on existing hardware systems. Thus a set of end-case operands would be set up for routines like ALOG and its vector counterpart.

All object library routines should use the data flag branch register for error flagging and the FORTRAN supplied data flag manager routine for reporting errors to the user. When vector functions encounter errors, an automatic system for rescanning the results to find the out-of-bounds results and corresponding input routines should be invoked so that the user is relieved from the burden of analysis.

The FORMAT, INPUT, OUTPUT, and DEVICE status routines usually involve a great deal of software "chit-chat" which implies many CALL sequence executions wherein no other useful work can be done. The FMP will utilize Backing Store to perform pseudo I/O for the normal production job. This will be accomplished with the concurrent Map Unit, and implied data moves using LEVEL statements. With the exception of FORMAT processing then, these I/O functions should be performed by in-line instruction sequences which can be scheduled among useful vector arithmetic operations.

Formatted Input/Output should be either performed on-line (that is by a function call to the cracking routines in the object code sequence) or off-line (transmission of data, pointers, and the raw format to the Backing Store where it is blocked up and sent to another processor in the system to be formatted in final form). In either case, formatted Input/Output requires further design analysis.

3.3.6 LINKING AND LOADING

All programs submitted to the FMP for execution will be delivered in a complete, prelinked and loaded binary form. This block-loaded form is called a "controllee file" and contains, in addition to the complete set of binary modules, tables describing the regions in each level of memory assigned to the program, beginning location of the Main Memory, Intermediate Memory, and Backing Store dynamic spaces.

Integral to the compiling system then, must be a "loader" function which can gather separately compiled modules with selected object library modules from a variety of inventory caches (files), link the data and entry points together, establish local and global working spaces for each module, and generate initializing information for preset data areas.

The loader performance is only critical when a large number of "compile-and-execute" jobs are passing through the system (during debugging of new applications, for example). Of much more concern is the existence of extensive diagnostics which the user can readily understand both at load time and execution time. In case of catastrophic failures (where even the best program goes berserk) some degree of audit trail should be salvageable from the contents of the various memories to help the programmer find his error. Each module should therefore contain, in addition to the executable binary code and data constants, a series of tables which are used by the loader to line and map routines into the controllee file, and which may electively be retained in the binary controllee file as an aid to debugging or error recovery.

An example of this type of system called the MODULE HEADER TABLE, is shown on page E-3 (appendix E). Each table begins with the ASCII name of the table, in this case "MODULE", that can easily be found on visual scan of an ASCII dump, or by vector scanning memory. The module length in 64-bit words is found in word 2 of this table. In word 3 appears the ASCII name of the module, usually the PROGRAM, FUNCTION or SUBROUTINE name. A time stamp for when the module was created appears in word 4 and the processor name and version number can be found in word 5. The header points to a series of additional tables (diagrammed in the remainder of appendix E), which supply loader information and debugging information for object-time debugging. Most of the table functions are self-explanatory in their name (a tutorial on the loader methodology will not be given here), but two tables deserve a little discussion -- Interpretive Data Initialization and Relocation tables.

There are two techniques for performing data initialization and initialization of relocation pointers at LOAD time. When dynamic loading is called for (load occurs at time of CALL) as might occur in some system routines, the initialization is done by a sequence of generated objct code which is caled the "executable data initialization or relocation table". For normal static loading of FMP applications programs, the complexity of some initialization is better handled by the loader interpreting table entries one at a time. Thus constants and relocation pointers can be scatter-gunned around memory by the loader, or loaded in nice sequential streams, depending on the needs of the code. Relocation pointers are addresses in the code (relative to the beginning of the code) where non-relative branches have occurred or pointers into the register file where static addresses point to code segments or local data quantities. These must be updated as the module is placed in memory following another module, and all addresses thus relativized.

The structure given here permits all of the object code to be aglutinized in a lump with only the two-word header "CODE" intervening between modules. The remainder of the tables may be kept in the Backing Store, with the CODE ponter set to point to its particular MODULE header in Backing Store. In the event of error conditions or debugging actions, the system can either locate appropriate tables by referring backward from the linked code, or locate the linked code by searching Backing Store for the MODULE header and then using the memory pointers to locate the needed information. This is particularly suited to the use of the DEBUG symbol table and SYMBOL definition table which are used by the symbolic FORTRAN DEBUG option on the FMP. With this option, execution can be breakpointed (halted on a particular form of reference to a symbol, including execution of labeled FORTRAN source statements), data can be examined or replaced, and formatted dumps with symbolic names produced. This feature is considered essential in a system as large as the NASF and must be integral to the design of the compiler and loading and linking system. Figure E-1 (appendix E) shows a mixed hexadecimal and ASCII dump of a small controllee file to demonstrate how the tables are usually allocated in memory and how data can be located by the programmer or an analyzer program.

3.3.7 OPERATIONAL CHARACTERISTICS

Another aspect of the compiler must be specified in the NASF procurement. This involves the execution characteristics of the compiler itself, its performance, code space, and compiling features.

This leads to one of the most difficult questions that has been addressed by the series of NASF studies, whereabout should lie and labor the compiler?

The problem with situating the compiler within the NASF is one of strategy and not of technical capability. The following discussion will reexamine some of the issues that have been discussed in previous reports, and in meetings with Ames personnel.

1) The development of a complex and yet stable compiler, plus supporting object library, is a lengthy and consuming process. If at all possible, an existing compiling system should be used on which to base the FMP FORTRAN in order to reduce cost, schedule, and reliability risks.

2) Until an FMP is operational, potential users and software developers will have to rely on existing computer systems to support programming, compiling, and debugging. The availability of a complete FMP FORTRAN system on these "interim" mainframes is highly desirable for the total NASF success.

3) Compiling on a front-end processor intead of the FMP might be a more effective use of the FMP, which is of course designed first and foremost for high speed arithmetic processing. Certainly the turnaround of compiler detected errors and code listings would be quicker when processed by the front-end processor than the FMP.

4) The specific architecture and model of the front-end processors may not be under the control of the FMP developer and may not be identified until late in the development cycle. It can be expected that the number and qualities of the front-end processors may change over the lifetime of the FMP. Certainly NASA may want the option of varying those parameters of the system as interactive workloads and front-end software features change.

5) "Cross-compilers", which operate on one machine compiling for another machine, have suffered in the past from the need for two machines with which to experiment and develop highly refined optimization tecniques that become ever more important in the maturing years of the target system.

At the outset of this projct Control Data recommended that the compiler reside on the front-end processors and produce code for the FMP (also called the back-end processor). In addition, it was suggested that the loading and linking function also reside with the compiler. As time passed it became obvious that the development cycle for such a compiler pointed toward use of an existing Control Data compiler for either STAR or 7600 as a base compiler. The STAR compiler recommended itself because it was structured to support expanded automatic vectorization as well as vector extensions. One of the reasons for dabbling with the STAR language as the FMP language arose from this rationale. Retention of the STAR scalar instructions, addressing schemes and I/O interface schemes gave weight to the possibility that the STAR compiler could be used in toto, with only minor extensions being necessary. When the FMP FORTRAN (described in Volume III) finally became firm, RADL realized that major changes would have to be made in any compiler to meet the requirements for the FMP. The "almost" free solution of compiling on the FMP for the FMP became no longer free and the "sitting" suggestion had to be reopened for the compiler again.

In the opinion of RADL, the optimum solution would be the development of a new compiler, written in a higher-level language such as PASCAL, designed to reside on more than one processor, and capable of cross-compiling. The compiler should produce object code which can be debugged and tested on either the front-end processor or the FMP. Full optimization for the multiple functional units would be a selectable option for the compiler. This would provide the flexibility of having compile capablity on all processors in the system. Another advantage of this would be reducing the need for FMP availability during its early checkout period to assist in the object code generation checkout.

This optimum solution has, as stated often, the risks of meeting the system implementation schedule.

Compiler performance should also be specified in terms of compile rates, at least those of the CYBER 7600 or CYBER 200 family compilers. Statements per minute for an average FORTRAN program and for fully optimized output from the implicit and explicit metrics should be required of any proposed compiler. Compiler space is not a problem for the FMP but could present difficulties when used on a heavily loaded front-end processor. The compiler should be limited to a fixed residence no greater than that which the current 7600 and other large scale systems FORTRAN compilers require today.

The FMP compiler can easily rely on current compiler technologies, with some special emphasis placed on automatic vectorization, sceduling of vectors, and allocation of vector storage. The only factor in the compiler development that needs to be given considerable attention is that of the sheer manhours and elapsed time to provide the exposure and testing of the compiler and object code prior to the NASF going into full production status.

3.4 OPERATING SYSTEM FUNCTIONAL CHARACTERISTICS

3.4.1 GENERAL

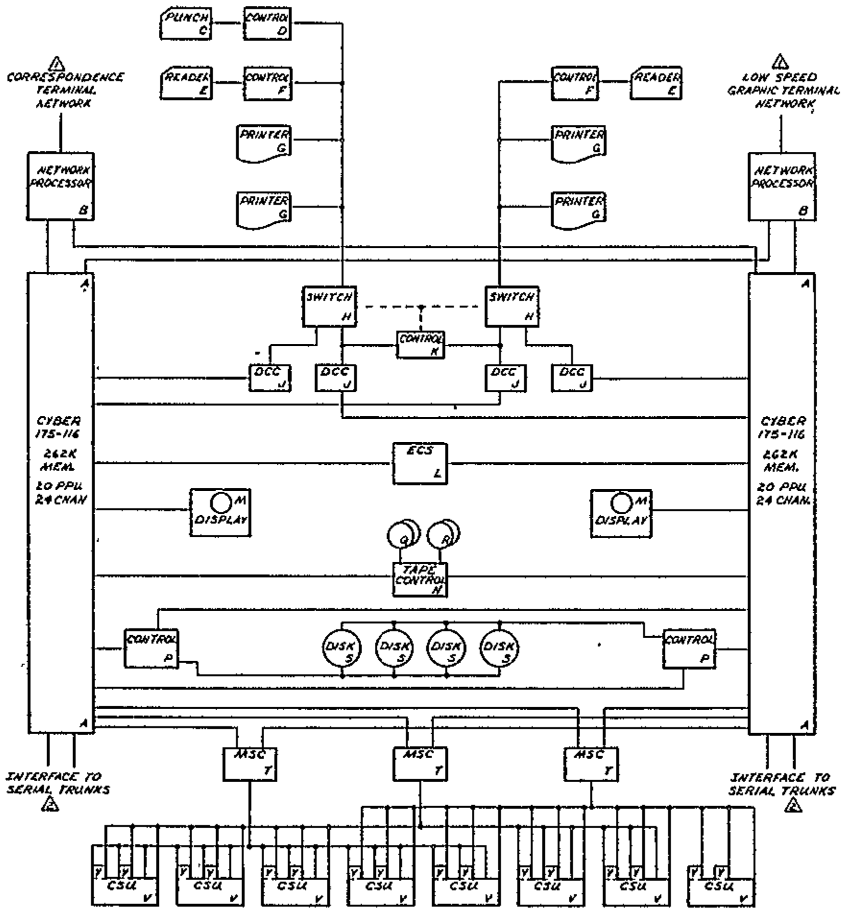
In preceding studies the NASF system concept and structure were described and diagrammed. The network of hardware that has finally been arrived at as an evolution from those studies is shown in figures 20 and 21. Three fundamental ideas have formed the basis for the NASF configuration and system software analysis.

1) Distribution of function among a number of possibly dissimilar processors -- A philosophy governing this distribution is summarized in the precepts:

    a) definition of system resource entities;

    b) management of system resources by intelligent (programmable) processors;

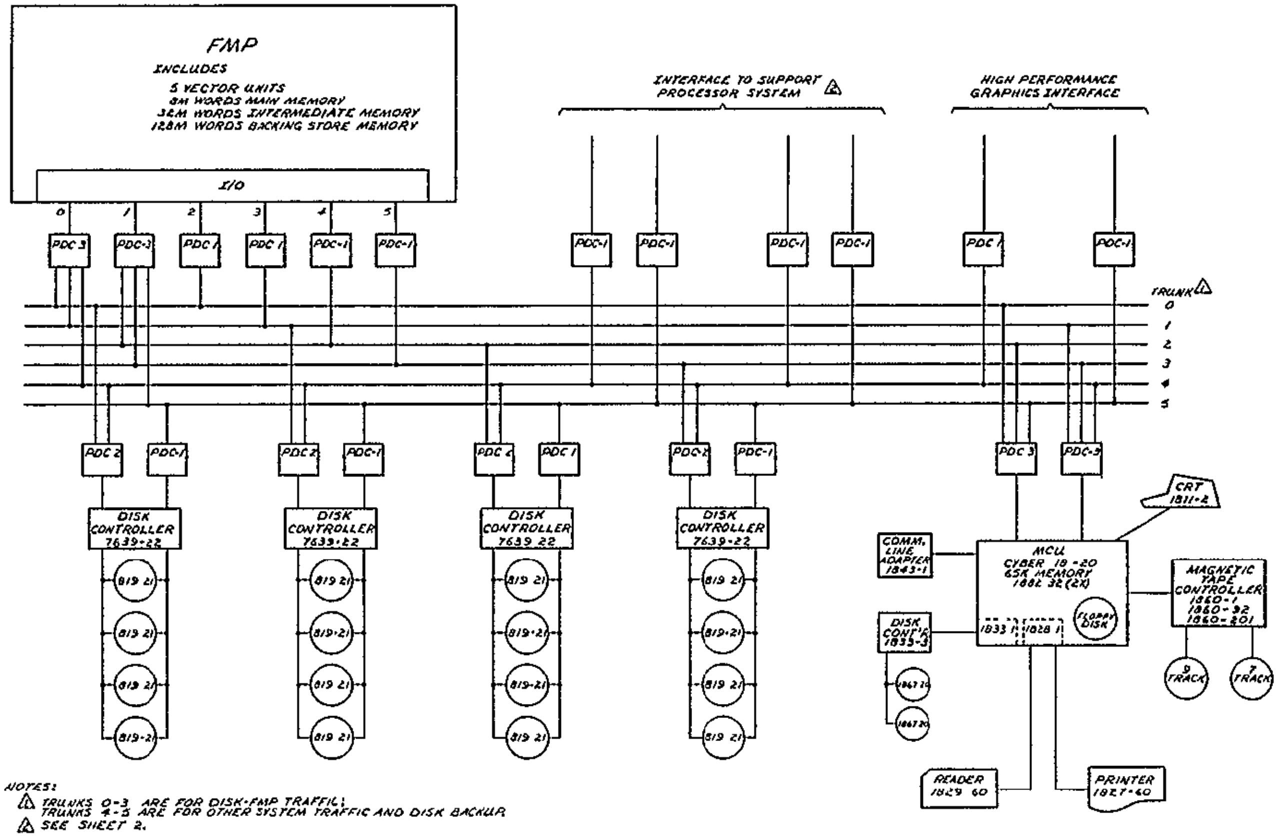
    c) proximity of resource and its managers should be as close as possible, physically, electronically, logically;



LETTER	MODEL#	DESCRIPTION
A	175-116	CYBER 175 COMPUTER
B	2551-1	NETWORK PROCESSING UNIT
C	415	CARD PUNCH
D	3446	CARD PUNCH CONTROLLER
E	405	CARD READER
F	3447	CARD READER CONTROLLER
G	580 200	TRAIN PRINTER
H	8271-2	CHANNEL TRANSFER SWITCH
J	10315-1/2	DATA CHANNEL CONVERTER
K	3270A	TRANSFER SWITCH CONTROLLER
L	7030-104	EXTENDED CORE STORAGE (524K WORDS)
M		DISPLAY CONSOLE
N	7021-32	MAGNETIC TAPE CONTROLLER
P	7155-1	FMD DISK CONTROLLER WITH 2ND CHANNEL OPTION
Q	677-4	MAGNETIC TAPE TRANSPORT, 7-TRACK (2 EA)
R	679-7	MAGNETIC TAPE TRANSPORT, 9-TRACK (2 EA)
S	885-12	FMD DISK UNIT (DUAL SPINDLE)
T	7880-1	MASS STORAGE CONTROLLER
V	7881-1	CARTRIDGE STORAGE UNIT
Y	7882-1	MASS STORAGE TRANSPORT

NOTES:
△1 INCLUDES BOTH LOCAL AND REMOTE TERMINALS, CONFIGURATION NOT DETERMINED.
△2 SEE SHEET 1.

Figure 20. NASF Support Processing System



NOTES:
1. TRUNKS 0-3 ARE FOR DISK-FMP TRAFFIC;
TRUNKS 4-5 ARE FOR OTHER SYSTEM TRAFFIC AND DISK BACKUP
2. SEE SHEET 2.

Figure 21. NASF Trunk Network with FMP

    d) resource management functions should be moved outward from a central computer toward the resource;

    e) "form follows function";

    f) processors with resource management functions have only knowledge (i.e., tables, pointers, etc.) of resources directly attached;

    g) each processor possesses a unique catalog of functions it can perform, all others are passed on;

    h) message discipline;

    i) common sense and reality must dominate any design.

2) Flexible interconnection of all components -- Using a new Control Data system called Loosely Coupled Network (LCN), all system components can be connected to each other using high-bandwidth, bit-serial data trunks which can extend for great distances.

3) A "computational engine" or highly intelligent arithmetic element in the total system -- The FMP should behave as a "slave processor" and not perform any system control operations. Its internal software operating system must be absolutely minimal in order that:

    a) FMP software development be minimized;

    b) other system software have a minimum number of interfaces to cope with in the FMP;

    c) the time needed in the FMP for system interaction processing can be reduced.

This last principle implies that the NASF will rely almost entirely on the operating system and system support software available on the front-end processors for management of the system resources. According to the "distributed system philosophy" the resources owned by the FMP are its functional units (Map Unit and Vector Unit) and its storage (Main Memory, Intermediate Memory, and Backing Store).

The FMP as a single entity is itself a resource that must be managed by some other processor. The management function has been delegated to the front-end processors or support processing system (hereafter called the SPS). The requirements for an operating system within the FMP are thus reduced to bare necessities. The extent of these necessities can be derived from an examination of how the FMP will be used in the NASF environment.

3.4.2 JOB FLOW

The following provides a recount of the probable sequence of events that will be involved in the processing of a flow model solution.

1) The user will, from an interactive terminal or batch input, initiate the execution of a model solution.

2) The SPS will select the already compiled, linked, and loaded solution program from the file system.

3) The contiguous binary stream representing the program and its locally defined data will be transferred to the 819 disks common to the FMP and the SPS.

4) The initial mesh data which has also been biding its time on a disk or archival file belonging to the SPS, will then be selected and moved to a block of storage on the 819.

5) External data and parameters other than the mesh information will be transferred to another set of blocks on the 819s.

6) A "message" (see reference 2) will be transmitted to the FMP from the SPS via the LCN and thence stored in the Intermediate Memory of the FMP. This message will contain the job description and pointers to physical disk addresses for each of the job components, binary program, mesh data, parameters, and location where output data is to be transferred.

7) The FMP monitor, in its own good time (when in idle loop, or when performing other system tasks), will discover the message in its queue.

8) The monitor will then schedule the transfer of all input data into either the Backing Store (if present) or the Intermediate Memory.

9) Each transfer is directed by a message to an attached PDC (Programmable Device Controller, the key LCN interface hardware).

10) When the transfer is complete, the PDC responds to the message.

11) When the transfer is complete, the monitor marks the job "ready for execution".

12) When the FMP becomes free, the monitor scans its list of jobs "ready" and selects the highest priority job to begin execution.

13) The new job is rolled in by moving its binary code into Main Memory.

14) Program execution is begun.

15) The FORTRAN program then reads it parameters, which is accomplished by map operations from Intermediate Memory.

16) Input meshes are also read in this manner, wherein an unformatted FORTRAN READ operation becomes a simple set of block moves of data from Backing Store to either Intermediate or Main Memory.

17) Processing is begun.

18) Formatted I/O is performed by subroutines in the object program, and data is transferred by means of map operations to buffers that are "psuedo-files". That is, there are no file OPEN, or CLOSE operations, nor I/O activity implied by READ and WRITE operations.

19) Upon program termination, PAUSE or interrupt all data in the Backing Store and Intermediate Memory buffers is rolled out to fixed positions (described by the initial job initiation message) to the 819 disks.

20) The SPS then associates each of the rolled out areas with actual SPS managed files to which the data is then moved.

21) The SPS processes the remaining job commands, either from the terminal or batch job, in order to reduce and display the result data, and to catalog or archive all other data that is to be retained.

In this scenario the 819 disks, the Backing Store, and Intermediate Memory all exhibit one common characteristic -- they are managed by physical address and stream lengths, not as files. Thus from a system standpoint, each storage medium is interchangeable with any other (albeit at significant performance differences). This makes it possible for the system to actually operate without two of the three bulk memories. In a minimum configuration for example, with constraints on problem sizes, the PDC connected to the FMP could intercept data storage and retrieval messages meant for the nonexistent 819s and convert them to block addresses and lengths in the Intermediate Memory.

This is possible only if the 819s are limited to containing transient data at all times and not permanent files. The advantage of such a scheme is obvious not only for purposes of degraded operation (while maintenance is being performed on the 819 or Backing Store, for example) but can be of great value during early system exercises where all the hardware need not be present. A phased installation of the NASF is thus facilitated.

3.4.3 INTERACTIVE OR BATCH?

The large amount of data usually ascribed to FMP jobs means that a single job roll-in/roll-out could require a significant amount of time. Interactive debugging usually implies many such roll-in/roll-out actions during the execution of a job. Should such activity be permitted in the FMP and supported by operating system functions? Despite the inherent inefficiencies that may arise from such a strategy, the potential impact on a high priority project that may need interactive debugging facilities could be enormous if this capability is not present. The "roll-in/roll-out" facility that must be provided in the FMP operating system to support the staging of data into and out of the FMP, can be used also to perform "checkpoint-restart" recovery dumps of the data base and code, when required by the system or the programmer. In addition, this same facility could serve in special circumstances to "roll-out" an executing job, when interrupted by the user, so that key problem parameters can be examined to determine if things are "going right" before allowing the job to consume a great deal of CPU time. Although the FMP is best used in a batch-mode only form, there must be some form of "escape-valve" permitted for privileged users (whose privileges are determined by the system-managing SPS) to STOP, PAUSE, and CONTINUE executing jobs. These privileges can have a serious effect on overall FMP efficiency, and therefore must not be granted lightly by the system manager or the operating system. The need for this capability cannot be overstated, based on the experience of the many large STAR-100 and 7600 user communities who make such demands of what would otherwise be a basically "batch-job" environment.

Another technique related to this question is that of a production job making intermediate result "dumps" which are to be sent to an output device (terminal, plotter, or printer) for evaluation while the program is still in progress. Although the system load model provided by Ames makes no mention of this requirement, experience with large code indicates that such a capability is highly recommended in the production NASF.

3.4.4 EXTERNAL CHARACTERISTICS

Given the basic requirements for job flow, what are the necessary external features as seen by the user from his terminal, and by the programmer in his FORTRAN code?

First, an aspect of the external characteristics of the FMP operating system must be discussed -- COMPATIBILITY. Since the majority of human interactions with the NASF will be with and through the SPS and other processors (such as graphics subsystems), the bulk of operating systems commands will be directed at those processors. If a function is to be invoked on the FMP, it should be described by the identical syntax and format as used on the SPS. More importantly, the relationship of such functions to the system should be the same. What does this mean? The Control Data CYBER operating systems are "file oriented". That is, data files -- raw binary and executable binary data -- are all retained in streams called "files". To invoke a system function or initiate execution of program involves the naming of the file containing the desired program. That file is opened, brought into memory and execution is commenced. At the conclusion of execution, the space used by that file is overlaid by the next file invoked.

Management of program entities is thus through the same file mechanism as is used for data. A control statement, whether submitted via a job control card, terminal or an executing program, thus consists of a file name followed by appropriate parameters. If the FMP is to be front-ended by CYBER machines then, this same relationship and command format should be maintained. The user then has only one set of concepts to assimilate and one set of formats with which to deal. The problem with this is that two effects can destroy what seems to be a nice principle:

a) A NASA choice of some other SPS, at the outset or later in the project, which has a completely different philosophy of operating system relationships and commands.

b) A change in operating system philosophy for the same equipment, a possibility, quite frankly, demonstrated more than adequately by the two Control Data operating systems for the CYBER family, NOS and NOS/BE. Although both are basically "file oriented", one adheres to that philosophy more than the other.

Only one solution can prevent the FMP operating system from being bent in the winds of change throughout its lifetime. That solution consists of eliminating completely any ability in the FMP for interpreting commands from the job stream, or to have any knowledge of resources outside itself (no concept of disk storage or SPS hardware permitted to invade the FMP). This is consistent with the distributed system philosophy, but not consistent with current system practices as, for example, in the "Symmetric Link" facility of NOS/BE. In this case, the back-end processor (such as the FMP) must "drag out" its own data base from the SPS through commands such as GETPF (get permanent file).

All command language processing is then performed by the SPS. The FMP and its 819s appear to the rest of the system as a block of memory to be loaded with data and programs and told to execute a specified program. Further, the user has access to FMP resources when executing on the FMP and thus the only external interfaces on the FMP are the FORTRAN language constructs given in the FMP FORTRAN Manual (Volume III). The I/O statements, PAUSE and END, provide implicit linkage with the operating system but no others will be countenanced.

3.4.5 INTERNAL CHARACTERISTICS

Having dismissed the subject of external characteristics so quickly, it would seem that the FMP operating system has vanished completely. There are, unfortunately, sufficient FMP management tasks that must be handled by the FMP to absorb the energies of a good sized development team. The FMP hardware contains some design features which are intended at once to constrain the creative expansiveness of such developers while also assisting their efforts in simplification.

a) Fixed memory allocations -- One 65,536-word block of Memory (beginning at address 0) and one 65,536-word block of Intermediate Memory (beginning at Intermediate Memory address 0) are set aside for the operating system. Job mode programs cannot access either of these two areas for program execution or data access. Monitor mode execution of code is limited to the 65,636 words of Main Memory, while the monitor is permitted access to all memories in their entirety.

b) Single interrupt -- The FMP can be interrupted having control transferred to the monitor by any one of the external PDC attachments to the I/O ports. While in monitor mode no other interrupts will be permitted.

c) In the event that a Backing Store block that is addressed is not present (because of reduced configuration) or the Backing Store is totally absent, an interrupt occurs, causing monitor to deal with the attempted Swap operation. If a Main Memory to Intermediate Memory map operation references addresses not present in the Intermediate Memory, a similar interrupt will occur. This provides a limited form of "virtual" memory to the FMP operation.

d) Monitor can establish upper and lower boundaries for data access to Intermediate Memory and Backing Store by the executing program.

e) PDCs have universal access to Intermediate Memory with the following restrictions:

   - Once FMP execution has begun after a deadstart condition, the PDC cannot access addresses 0 through 32,767 in Intermediate Memory. This is to prevent inadvertent destruction of the monitor kernel.

   - In deadstart mode the PDCs can write everywhere in Intermediate Memory.

f) All addresses in user programs for Intermediate and Backing Store Memory are relative to address 0 of the user space which begins at address ILBOUNDS (intermediate lower bounds) for Intermediate Memory and BLBOUNDS in the Backing Store. Both bounds are assigned by the operating system when the job is scheduled for execution. User addresses in Main Memory start at address 65,536 and are not relative.

g) Any attempt to read or write outside the prescribed bounds will result in an immediate interrupt to the monitor.

3.4.6 MANAGEMENT TASKS

There are a number of tasks which must be undertaken solely by the FMP itself because it is in the best position, or only position, to make judgments about resource utilization.

3.4.6.1 STORAGE

a) Allocation of one, two, or three job buffers in Intermediate Memory to permit the overlap of one job execution with the swap-out of another -- the allocation scheme must consider job size, priority, and amount of swapping required.

b) Allocation (and setting of bounds register) of Intermediate Memory for the job going into execution.

c) Presetting of Main Memory and Intermediate Memory to canonical NULLs (for security reasons as well as execution consistency).

d) Allocation of Backing Store Memory (and establishing operating bounds for the job in process) if that element is present.

e) Processing of illegal storage access requests by the user program, determining the nature of the request and transmitting the error information to the job's error processor.

f) Processing "storage not present" interrupts, moving data from alternate storage devices as required by the hardware instruction (map) that caused the interrupt.

g) Degrading the software storage maps when directed by the maintenance function to reduce memory so that maintenance actions can be performed on-line.

3.4.6.2 PROCESSING

a) Entering JOB requests into internal queue in priority order.

b) Aborting an executing job or deleting from queue on demand from front-end processors.

c) Scheduling and execution of on-line "confidence" diagnostics.

d) Initiation of job.

e) Processing of job calls for systems services.

   1) Transmit message to SPS.
   2) Explicit Input/Output.
   3) Wait on external message.
   4) Terminate job normally.
   5) Abnormal job termination.
   6) Initiate checkpoint/restart.
   7) Request job accounting information (date, time, time on, etc.).
   8) Increase or decrease storage allocation in any level memory.

f) Trigger job roll-out by PDCs.

g) Linking FORTRAN "pseudo files" (TAPE1, TAPE2, etc.) with physical memory space in Backing Store, Intermediate Memory, or local RMS (rotating mass storage).

h) Executing special system functions requested by the Maintenance Control Unit.

i) Transmitting job statistical data to Maintenance Control Unit.

j) Transmitting exceptional condition status to Maintenance Control Unit.

k) Management, reorganization, and culling of processing queues.

l) Deadstart system initialization.

The fairly long list given here is not as complicated as it might seem since, in most instances, the FMP monitor consists of a set of privileged subroutines each of which is called either by the job in execution or by a job operating in the SPS. Thus the SPS message "ADD JOB", invokes a single monitor mode subroutine in the FMP which inserts the incoming job into the execution queue, with appropriate "time stamps" and then returns control to the executing job. "Time slicing" is not permitted for FMP jobs, and roll-in and roll-out are controlled by the SPS program called the FMP manager. The FMP manager has a total responsibility for the optimization of use of the FMP resource. Decisions regarding how much memory at each level to allocate for a given job are done in the SPS. The allocation message is then sent to the FMP monitor which performs the actual allocation act (by entering data in appropriate tables and setting internal registers when required). So the functions listed above are, in fact, very primitive "slave" operations which behave more like "drivers" of the hardware switches and registers in the FMP.

The FMP "Operating System" therefore exists as a job on the front-end processors, which make all decisions about allocation and scheduling. This makes it possible to begin development of the FMP Operating System on an SPS as soon as the final configuration has been decided upon. The FMP manager would definitely be written in a higher level language (such as PASCAL) and would operate as a normal user job on the SPS. It would deal with the FMP as if it were a piece of peripheral equipment, whose controllable functions were only slightly more complex than start, stop, transfer data, report errors.

3.4.7 PERFORMANCE CRITERIA

In the specification of an operating system for the FMP, some measurement points and criteria should be required for degree of functionality, storage utilization efficiency, and processing efficiency. RADL has already forced a limitation on space utilized by the FMP Operating System itself by hardware design "fiat". Some similar criteria should be established for the FMP manager on the front-end processors. Maximum memory residency on the SPSs should be limited to 32K SPS words, so that sufficient space remains for the other system management functions on those machines. The FMP manager must have its critical kernels resident at all times in the FMP to handle the processing load expected and to meet the response time requirements.

If a full queue of completely vectorized jobs in the FMP is assumed, then it can be required that the FMP monitor perform the servicing of such a job stream with no more cost than a 5% burden of additional elapsed (wall clock) time over the theoretical time to complete the same set of fully vectorized codes. This means that the Vector Units would be kept busy 95% of the total machine hours (assuming the compiler has 100% vectorized the codes) in an operating day.

Minimum system response times for functions performed outside the FMP can be established for each message type listed above. For example, in the case of "explicit I/O" requests, the time taken away from the job for the monitor processing of the I/O message must be limited to some infinitesmal period like 20 microseconds (during which time the Vector Units could have been performing up to 60,000 calculations except for the interruption). Further, the time required to service the request must not be greater than 5% of the theoretical service time. As an example, take a disk read request of 65,000 words from the 819. Given the average latency and transfer times, the opeating system additional time for the transfer cannot exceed 5% of the theoretical total of latency and transfer time.

In the same way, criteria can be established for response times from the SPS, maintenance units, and other attached processors, for each function listed in the preceding discussion.

Finally, the FMP manager must be bounded by minimum performance standards. Assuming an available backlog of jobs flowing into the system, the FMP manager must keep a minimum queue of 3 to 5 jobs built in the FMP (which then of course, must guarantee efficient processing of that queue). Response times for servicing of critical messages, such as interactive display and interrogation of the FMP functions, must be established to keep the FMP fully busy. If the FMP manager in the SPS cannot respond quickly enough, then it must at last respond to the FMP with a "roll-out" function so that the FMP can begin arithmetic processing as soon as possible.

Final specification of these attributes of the FMP Operating System can be done as more experimentation with the system models and Ames load data point out the critical spots in the network that must be controlled during system development. It is expected that work with these models will continue into subsequent phases of this project so that a sufficiently rigorous specification can be produced to guide the implementation of the FMP Operating System.

4.0 FLOW MODEL PROCESSOR SIMULATION AND ANALYSIS

4.1 FMP SIMULATOR

Before construction of an FMP can begin, Control Data believes that the entire assemblage must be designed and simulated at the fundamental circuit level. This simulation would necessarily involve the execution of diagnostic sequences (to determine functional adequacy) and the execution of key portions of the performance metrics (to determine the execution speed). The design process for Control Data super-computers also includes simulation as a tool from the very inception of the effort through actual machine checkout. Even with the most powerful computers available on which to execute the simulators, the number of logic events simulated consumes more computer time than is reasonably available.

It becomes necessary then to establish a hierarchy of simulation systems, each one capable of a certain level of analysis of the design and FMP machine operation. The three levels identified are:

a) "Gate level" -- This simulator is used by the hardware designers to check the behavior of the actual circuits and wired interconnections with a very high resolution. In most design cases the resolution of the signal analysis is carried out to 50-picosecond intervals. This level is used to verify design before parts are ordered for the machine.

b) "Block level" -- This simulator represents aggregates of several individual components as "blocks" of logic, for example, an adder block capable of performing a two's complement add operation on input operand pairs of 16, 32, or 64 bits, or any increment between 16 and 64 bits. These blocks are then interconnected by the designer with the same diagnostic and performance sequences then run through them as for the gate level. The execution speed on current machines for this level can be 10-60 times faster than the gate level, thus larger assemblages of logic can be simulated over longer sequences of input instructions. The processing of a single vector operation on the FMP, however, still consumes extraordinary amounts of time even at this level of simulation.

c) "Functional block" -- Many events in the preceding simulation occur at the picosecond or nanosecond level. When one tries to examine the behavior of an overall performance metric though, the effects of memory data transfers become important to the final performance estimates of such a machine. In a simple case, the moving of a single slab of data from Intermediate to Main Memory using the Map Unit requires (for the

> implicit example) that 65,536 words be moved at the rate of eight words every clock cycle for a total of 8192 clock cycles. Performing analysis of this activity at the block or gate level is prohibitively expensive in terms of simulator time, and submerges all of the more refined circuit and block analysis, as far as elapsed time is concerned. A third level model is then desired which allows interaction of all functional units to be modeled at less resolution for longer periods of time.

This approach is quite similar to the actual design process. First the architect draws out the overall scheme and fragments it into logically separable entities (functional blocks). Then estimates are made as to the performance and functional requirements of each entity. A computerized model is then developed for each entity, and the lot subjected to some form of programmatic analysis. If the architecture holds up, the design is initiated. Each entity is then broken down into a set of constituent logical blocks and this breakdown modeled to determine if all data and control paths are in place and timed to the nearest clock cycle. Finally, each logic block is broken down to circuit elements and coax and foil interconnect, and the entire assembly is "tuned" to the nearest 50-100 picoseconds. Manufacturing documentation is then produced automatically from all levels of the simulation; the parts design is extracted from the gate level model, put on tape, and shipped to electronics vendors. The machine is then constructed.

At each stage of the design, the higher level model is verified against the continually refined, more detailed models. For example, the functional model of a vector unit would have a sequence of vector instructions put through it and timings would be analyzed. As the block level design is completed for a vector unit the identical sequence is put through the more detailed model and the timings compared to the higher level model. Where necessary, the higher level model is adjusted to reflect the actual design. This is carried out once again as the gate level design and simulation is carried out, with the more gross timings of each higher level model matched against the actual hardware design. In a well structured design environment, all three models use the same interfaces, control cards, and file structures so that a "mixed" simulation might be executed where the memory system and scalar unit can be simulated at the functional level, the vector stream control at the block level, and a single vector unit simulated at the gate level. This interchangeability permits the designers and potential users to evaluate various critical networks and performance questions without having to run the entire ensemble at the gate level.

To carry out the FMP design for this study period of the NASF project to a degree that ensured the buildability and performance of the system, all three levels of design had to be explored to some degree. A modest amount of gate level design was undertaken to answer some design questions not addressed by LSI design already in progress at Control Data. A good deal of block level simulation was done for the memory system (since that is the crucial element in the FMP bandwidth) and the vector units (since they perform the useful work and take the major amount of logic in the FMP). To the extent that detailed design has been carried out, the block model represents the actual behavior of the FMP to the clock cycle level. This model was first developed for Ames in the second study period of this project and reflected the design of the FMP at that time. Throughout the summer of 1978 this model became known as the "Detailed FMP Simulator". As the FMP architecture changed from that described in reference 2, the simulator underwent major changes, some structural and some procedural (input, output, and use). Versions of this simulator have been delivered for use by Ames, and with this report (Volume IV) the simulator is formally delivered to NASA.

The functional level simulator has been called the FMP High-Level Simulator, and is documented in Division 4 of Volume IV of this report. For significantly long code sequences where vectors are long (100 or longer), the high-level simulator is an accurate representation of the current FMP design. Where short vectors or single element map operations are to be analyzed, the detailed simulator is required since only at that level are specific memory conflicts modeled. The current detailed model has not been completely verified against the latest design changes in the Map Unit so it is possible for short vector data to be as much as 10% in error from what the actual hardware would yield.

4.2 BENCHMARKS FOR THIS STUDY

Divisions 2, 3, and 4 of this volume contain discussions of the four codes submitted by Ames to Control Data as metrics around which the FMP was to be designed and analyzed. The characteristics of the implicit code--the number of computations to be performed can be estimated exactly for any set of input parameters-the code appears to be the best candidate for long term production on the FMP--have led it to be the primary focus for all hardware and software design efforts. In previous reports the characteristics of this code have been explored and the computational behavior analyzed to help direct FMP design efforts.

The weather codes were of great interest at the outset of this phase since the question had been raised as to the suitability of the FMP to broader applications areas. It was determined that for the two weather codes provided, the original FMP design (as documented in reference 2) would not achieve the 1000 megaflop goal for the FMP.

Four different approaches were used to pursue the benchmark analysis. This was due, in part, to the experience and specialties of the staff working on the codes, and in part because of the very nature of the investigation.

1. The implicit code became the central theme of the FMP design and the language and compiler specifications. As a change was introduced in the hardware or software, the implicit code, or a portion thereof, was used to test the idea. The simulators were first exercised with the implicit code. A decision was made to treat the entire implicit code with respect to language, compiling, and simulation in order to ensure that some design feature was not overlooked.

2. The explicit code employs solution techniques which differ in some ways from those of the implicit code. The methodology requires convergence of the solution, a data-dependent feature which controls the number of calculations, unlike the fixed number that are inherent in the implicit code. Apparent recursion relationships impact the amount of vectorization that can be readily attained without severely restructuring the code. It was decided to turn over this evaluation to a new group of analysts and have them attack the code without predisposition toward the FMP. They were instructed to vectorize the code as directly as possible from the statements in the original source language, and make the code operable on the STAR-100. The conversion of the STAR-100 code could then be done fairly mechanically to the FMP, since the primary functions used would be map and arithmetic operations. The answers obtained on the STAR-100 would be checked to make sure the algorithm conversion was "complete" and then simulator input for the FMP detailed and high-level models could be transliterated from the STAR code.

   With the limited analyst resources available, it was decided to tackle only the LX, LYC, LYI, and CHARAC routines for the purposes of recoding and simulation. The LX and LYI subroutines were vectorized in a direct manner, rewriting local DO loops only. All data passed between these subroutines and the remainder of the code was left in the original scalar allocation so that the code could be run with and without each of the vectorized codes, on the STAR-100, to obtain correct answers. This technique led to object code for the FMP with small vector lengths, compared to the implicit code, but the exercise did provide a vehicle for testing the performance that could be gaind by making simple first-attempts at vectorization. The

TURBDA subroutine and LYC turned out to be vectorizable without much thought or effort but CHARAC proved to be the challenge to vectorization. Initial efforts at converting the code left most operations still in scalar mode with a performance of only 27 megaflops. By calling CHARAC from LYC and LZC to process all data in the I-K plane at one time, CHARAC could be vectorized using the control vector capabilities of the FMP. Estimation of performance could not be done using the simulators, because the degree of sparsity of the permissive bits in the control vectors has a direct impact on the gigaflop rate. Making the assumption that for every pass of three there was a reduction by half in the active calls to CHARAC (represented by the control vector), performance of CHARAC was estimated for the 100x100x100 mesh case at about 200 megaflops.

3. The spectral code which originated at MIT was analyzed to determine what special machine characteristics were needed to make it meet the performance goals of the NASF. The FFT and SPCFOR (Legendre transform) routines were chosen from the spectral code for simulation because they constitute the computational heart of the spectral code. These routines were vectorized quite directly and simulated for two problem sizes. The first problem, with relatively low resolution, represented 25 layers of atmosphere, 6 wave numbers and 15 latitudes. A more reasonable form for useful production work, in the opinion of CDC investigators, would consist of 48 layers, 21 wave numbers and 53 latitudes. The two cases were simulated to show the range of performance for this code.

4. A shortcut method of handling the GISS model was tried in order to reduce the resource commitments for this secondary phase of the project. The GISS model had been fully vectorized for the STAR-100 and a copy of that model was available to us. If necessary any segment of that code could be transliterated from STAR to the FMP and the results subjected to either detailed or high-level simulation. Instead, Computer Sciences Corporation (CSC) continued to project FMP performance on the codes using the simulator delivered with reference 2. As a result of that study, the FMP was shown to be substantially lower in performance on the

> GISS model than RADL investigators assumed, based on the performance of the fully vectorized version on the STAR-100. One example unearthed was a routine called LINKHO in the GISS model which ran at rates far less than 100 megaflops, and was allegedly unvectorizable. Shortly after, a corresponding vector portion was extracted from a current STAR-100 version and simulated without change on the FMP, vintage 1979. The result rate was shown to be 700 megaflops for that one routine! The LINKHO and AVRX routines were thus chosen for further study, with a representative portion of LINKHO being simulated with the LVL1 simulator for the FMP. AVRX was chosen for manual extrapolation because it represented the physics computations of the GISS code and because its vector characteristics could be estimated by hand.

Mention of this difference is not meant to reflect on CSC's effort, but highlights two important difficulties in this evolutionary effort. The FMP has been a moving target as far as design is concerned since the inception of the project, and thus estimates made in summer, fall and winter of 1978 would vary wildly. The second problem is that "vector programming" or "vector thinking" is still a young discipline and still functions more as an art than a science. In this case, the author of the original code was able to see and recognize the "parallel" nature of some of his own constructs and thus make certain vector judgments in the GISS recoding, whereas a new analyst, confronted with a code not of his own making and a machine which is a bit alien, would find great difficulty in his first attempts at mapping code.

Neither of the weather codes taxed the FMP memory system as did the implicit and explicit codes. In fact they can be completely contained in the Main Memory of the FMP. The major concern in achieving performance on these codes is the degree to which scalar operations have to be employed, where no overlap of scalar and vector can be done, and the lengths of vectors. Simulation results for some segments of the weather codes achieve only 100-300 megaflops because of the presence of short vectors and the effect of vector startup time. The hardware design attributes that contribute to startup time have been discussed previously (in the hardware design section of this report), and relate to memory access time, pipeline lengths, and interconnections. The detailed FMP simulator imposes an average of 6 clock cycles per startup of each arithmetic vector. (Design analysis to date indicates that 6 cycles is a reasonable expectation for average vector startup.)

It can be seen that if vector startup time were zero and vectors could be scheduled back-to-back by the compiler, then the actual computation rates would in fact be the same as the theoretical rates. Thus rates of 500, 1000, and 1500 megaflops could be achieved on the weather code. If vectors are processed which are equal in length to the number of levels in the model, then lengths of 16-32 would be expected. Given a 1000 megaflop peak rate for such lengths a table of statup costs would show:

STARTUP CYCLES	MEGAFLOPS (vector lengths of 16)
0	1000
1	666
2	500
3	400
4	333
5	285
6	250

In truth, the Control Data FMP can have zero vector startup for many types of vector operations, in particular when there is no "loop back" in the pipelines between functional units, or mode changes for the data paths in the Vector Unit between vector operations. It is possible then for a carefully coded (yet brute force) conversion of the weather codes to achieve close to the 1000 megaflop threshold.

4.3 FUTURE METRIC STRATEGY

One of the significant outcomes of the FMP investigation has been the realization that the tradeoffs between memory hierarchies, processor organization and interconnection, and problem statement (including data structuring and movement scheduling) create complex ensembles about which some decisions must be made. In particular, the FMP structure proposed by Control Data offers three levels of memory, with consequent levels of bandwidth, combined with functional parallelism which provides concurrency of operation based on the separation of functional identity.

A brief examination of this multi-faceted architecture in the face of the problems expected to be applied to the NASF indicates the tremendous variety of performance that can be achieved. The small problem, which could be used as a research tool or for debugging a new set of parameters, will of course fit in the Main Memory and operate at much higher map rates while possessing shorter vectors that have commensurately lower throughput rates (due to the effect of the spectre of vector overhead). On the other hand, certain research codes may exceed the LEVEL 2 memory and need to operate at lower rates than 1 gigaflop, in order to achieve the goals of the research project. The NASF must be able to accommodate the entire range of programs in order to be a viable research and development center, as well as a high capacity production facility.

The meaning of these observations is clearly that the set of "benchmarks" or "metrics" that are used to evaluate prospective computer systems for the FMP must be variegated in quantitative as well as qualitative spectrum. Thus there should be a two-dimensional approach to the "metricization" of evaluation programs for the FMP.

One dimension of this measurement system should be the type of processing. Thus the implicit and explicit aerodynamic codes should be the primary evaluation tools for design, simulation and actual installation of the NASF. The interest and investment in the weather models should then, of course, require the inclusion of characteristic approaches to the weather projection problem. The two such models, spectral and finite difference, are offered by NASA as representative of two "poles of thought" in weather investigations. Two other important models that were not investigated, and to which it can easily be forecast the NASF will be subjected, are the "particle in cell", or in another context "Monte Carlo technique", and the "structural" codes.

Considering the massive investment in financial and human resources, as well as the extensions into other disciplines that are planned for this facility, it would seem that a rationalized and systematic method needs to be employed for the analysis and evaluation of the NASF system to be procured. A suggested methodology for this might be:

a. Identification of type

   Implicit Navier-Stokes, time-averaged, algebraic turbulence;

   Explicit Navier-Stokes, time-averaged, algebraic turbulence;

   Microscopic, full Navier-Stokes turbulence model;

   Finite difference weather circulation model, operational forecast;

   Spectral solution, weather circulation model, operational forecast;

   One weather model for research purposes (mathematical method is optional);

One existing structures code, finite difference;

One particle in cell model.

b. Identification of size

Configurations should be chosen for each type that represent

1) an existing configuration that can be run on and compared with numerical results and performance of existing systems (such as the CDC CYBER 76);

2) two typical configurations expected for everyday production work (the variance of any one of the three dimensions in the implicit code, for example, can have a substantial effect on performance);

3) a maximum production run that is conceivable for the 1984-1989 timeframe;

4) a realistic research model configuration.

(For example: The existing flow models are presently running on the CYBER 76. These should be the baseline mesh sizes for these models. The expected 50x80x80 mesh should represent the production data base sizing, and the 100x100x100 should represent the MINIMUM THRESHOLD of performance benchmark. Finally, a research model of at least 500x500x500 should be chosen as a "metric" configuration.)

c. Prioritization

The designers proposing a NASF system must be given as many degrees of freedom as is possible, given the massive national attention that will haunt this project. Therefore it is imperative that a system of evaluation be established, AND ADHERED TO...NO MATTER WHAT THE PRESSURE, which

1) picks the main emphasis of the NASF (thus the minimum performance should be pegged at one gigaflop for the 3-D implicit code);

2) establishes an acceptable mix of use of such an implicit code for 1) test cases, 2) production, 3) research models;

3) identifies the relative priority of interest in the other codes (explicit Navier-Stokes, weather, structural, etc..);

4) provides measurement specification.

The implementation of the NASF will proceed through three more distinct steps:

1. Proposal of candidate architectures.
2. Design of candidate architectures.
3. Construction of final architecture.

The set of metrics that are arrived at for evaluation must be subjected to performance analysis at each of these phases to determine the viability and reality of a given design. In the first instance (proposal), all parties offering candidates must be requested to provide estimates of performance broken down by section of probable machine code. In the second instance, the analysis should consist of systematic breakdown and simulation of the same portions of the code that were analyzed in the proposal phase. This simulation should be accomplished at the detailed level of the design to verify that the hardware to be built will, in fact, meet the proposed performance objectives. The cost of such simulation is such that a complete simulation of a metric run is impossible. Instead, the candidate design simulations should be allowed to make segmented analysis, without requiring all trips through all loops, with a specification as to how the single-pass results may be extrapolated.

The implicit model which has been discussed previously may be used as an example. A detailed design simulation is required of

1) the first trip through each sweep loop,
2) a subsequent pass through each sweep loop,
3) the last pass through each sweep loop (to cover any end cases), and
4) one trip through VISMAT,

in order to get the basis for detailed estimates of the left-hand side computation. Given the sets of KMAX, LMAX, and JMAX taken from the sizing task (b. above), these simulation results can be extrapolated for each sized case. The method of extrapolation should be specified by the consumer (Ames) so that a uniformity of approach is guaranteed from all candidate design offerings.

Finally, each metric in each size must be prepared for execution in full on the final operational machine. In addition to the reliability, confidence, and other specified acceptance tests, this set of metrics should become the final acceptance test for performance of the FMP.

Note that for some of the metrics (particularly the Navier-Stokes research models), a data flow analysis between memory hierarchies is required for each phase of the design and construction effort, as well as the computational performance of the FMP. This data flow will include not only backing storage modeling, but I/O transfer and support processing system activity as well.

All evaluations of prospective FMP systems should then start with the absolute requirement to achieve the 1 gigaflop level for test cases, production runs, and 100x100x100 maximum configurations. The remaining cases must be coded, simulated and estimated for their individual performances without a mandatory performance level being required. All prospects then are given the optimum level they must obtain for the primary mission stated for the NASF, but also must provide data relevant to other possible uses.

After this phase is concluded, the advice of all contractors should be collated to form the quantitative basis for the other aspects of this evaluation.

A major dilemma facing Ames reseachers is the decision that must be made in the event that one candidate could be shown to perform substantially greater (or substantially less) than all other candidates for all codes other than the implicit, 3-D, aerodynamic code. A weighting factor then must be applied to the performance of the NASF which not only includes probable mix results but also NASA-Ames priorities. It is folly to think that such priorities can be identified strictly by money or its equivalent in machine time. There are, for such a national facility, many more considerations which must be identified and prioritized by NASA before a meaningful competitive analysis can be commenced on candidate designs.

4.4 BENCHMARK SPECIFICATION

In the preceeding section there is mention of the problem of properly specifying the performance benchmarks to be used for procuring the NASF. The problem arises from the scenario which is believed to be the best approach to designing and evaluating the FMP.

Specification of at least two production codes representative of the anticipated NASF workload.

2. Specification of at last one code representing the research environment which will share the facility.

3. Specification of at least four additional codes which demonstrate computational behaviors different from those exhibited in the preceding three codes.

4. Specification of a user-produced full-function evaluation. (This would demonstrate to NASA that all specified functions are present in the FMP and operating according to the documentation. These would be in addition to the vendor produced functional diagnostics.)

5. Creation of the FMP and system, and production of high-level simulators for both.

6. Execution of 1 through 4 in their entirety on both the system and the FMP simulators.

7. Derivation of the block level design of the FMP, and production of the block simulator.

8. Execution of 1 through 4 on the block model to verify functionality and performance levels.

9. Detailed design of the FMP and production of the gate level simulation of the FMP from that design.

10. Execution of parts of 1 through 4 on the gate level model.

11. Negotiate price and schedule.

12. Order parts and build.

13. Demonstrate 1 through 4 on the actual hardware. Real execution must match the three levels of simulation performance within a prescribed allowance: no variance at the gate level permitted as measured by clock cycles, 10% variance from block level model due to unseen conflicts, and 15% variance from high-level model. These variances are similar to those that could be seen comparing the gate model with the block model and the high-level model.

This process would minimize the risks to both Control Data and NASA, and in effect, uses the simulation model as a "meta prototype", obviating the need for the construction and demonstration of an expensive and superfluous hardware prototype. Under no circumstances should NASA anticipate entering purchase agreements for a system of this magnitude without insisting upon a "fly before buy" demonstration. The demonstration can be valid using the "meta prototype" if the simulation systems can themselves be validated with real results. Each stage of the FMP development --high-level, block and gate-- can be evaluated before the next stage is initiated, thus reducing all partner's risks on entering the next stage.

As previously discussed, however, the perfect scenario is marred by the fact that all levels of simulation absorb more computer power than can be economically, or even practicably realized. The project is faced with the need to scale, first the actual production runs, and then the runs that can be made at each level of simulation. A summary of recommendations might be:

1. The base set of benchmarks should be codes now in existence, which are in operation on the 7600. Timings for each benchmark on the 7600 should become a baseline for performance comparisons.

2. The base set should be scaled in terms of mesh sizes, time steps, and iteration counts for at least three levels of probable production runs - small (perhaps 7600 size), medium, and large. The medium size might represent the normal workload parameters and the large might represent the extraordinary "research" models.

3. The base set should be representative of existing and projected computational requirements in functionality as well as complexity.

4. The entire base set will be run for all sets of parameters as the key acceptance test on the actual FMP hardware.

5. Each benchmark will have key segments extracted for simulation by the high-level simulator. These segments can be entire code sequences limited to one pass through a loop or nested loop. A multiplication factor for the number of trips can then be manually factored at a later time. Where code sequences contain branches that cannot be vectorized, or where vector contents and lengths are dependent on the opeating data, several different passes through the sequence exercising the strategic (in terms of performance effects) alternatives should be specified.

6. Sequences of code from these prescribed segments will be identified as "validation code". These sequences will be run on the block model to verify the timing assumptions used in the simpler high-level model. These will normaly be limited to mixed strings of instructions up to 25 in length consisting of scalar, vector, and map operations. Vector lengths will be limited to 100 or less.

7. Sequences of the code extracted in 6 (above) will be further distilled to provide gate level validation code. This is done primarily by reducing vector lengths and permitting some manual multiplier to be used to extrapolate operation for lengths of 100, when needed.

8. These last sequences will be run on a gate level basis to validate the performance at the gate level of individual units. Using mixed simulation with all other units operating either at the block or high-level and the unit under study runs in gate simulation mode.

So it is then that not only must the codes be sized, but judicious choices must be made of selected segments to be used for validation of the various simulation levels. This process has already been engaged during this study period as it became evident that the GPSS models required considerable time to run. Code analysis has consisted of breaking up the programs into sequences and then manually computing the combined operation timing.

5.0 PERFORMANCE ANALYSIS AND EVALUATION

This section addresses the methodology employed to evaluate the FMP design when subjected to the four metric codes -- 3-dimensional implicit, 3-dimensional explicit, spectral weather, and finite difference weather models. The major approach for all four codes was to derive performance data from runs of 'pseudo-compiled' code using one of the two simulators developed by Control Data for NASA-Ames. Several factors acted to limit the extent to which this was possible.

1. The machine time required to run large code sequences at the detailed (and more precise) level became prohibitive in some cases.
2. The number of FMP machine instructions which can be handled by the higher level (less detailed) simulator is limited to 2000 individual instructions. Loop structures are unrolled into linear code by the simulator and thus this limit of 2000 instructions constrained loop parameters in many cases.
3. The amount of code that could be pseudo-compiled exceeded the available time and resources allotted to this project, due primarily to the priority assigned by Control Data analysts to the detailed analysis of the three-dimensional implicit code. The effect of this decision was to absorb considerable time and talent in the reconfiguring of the implicit code, recompiling, and resimulating of the governing elements of the implicit code.

In many instances then, it became necessary to use extrapolation techniques based on simulator results and reasonable conclusions about the behavior of each unique code sequence. The manner in which the four codes were each addressed is discussed below.

5.1 THREE-DIMENSIONAL IMPLICIT CODE

The routines BTRI, VISMAT, VISRHS, MUTUR, and the FILTRX/AMATRX, FILTRY/AMATRX, FILTRZ/AMATRX segments of the STEP routine were simulated and analyzed for a variety of vector lengths, to determine the sensitivity of floating point performance to the length of arithmetic vector operations. This was of paramount concern because of the well-known effect of 'vector startup' time on 'short vector' performance in machines like the CDC FMP.

It was discovered that detailed level simulation of BTRI would have to be limited to processing sequences of code taken from BTRI, summarizing the simulator results, and then manually computing the probable behavior of BTRI in its entirety.

The same was true for FILTRX, FILTRY, and FILTRZ when simulation at the lower (more detailed) level was desired. It became obvious that the detailed model of the FMP yielded results within 1% of the high-level model when map operations (and hence unavoidable memory conflicts) were minimized. This was true in BTRI, and thus a simulator run was made for the complete BTRI using the high-level model; these figures are used in subsequent tables.

The heavy use of map operations to perform the tranpose and gather operations from Intermediate Memory to Main Memory in FILTRX, FILTRY, and FILTRZ required one simulation pass of these routines at the detailed level and comparing the result with a similar pass made at the higher level. For the vector lengths given later, the difference between simulator results in the high-level model and detailed model was ten percent or less. In all cases the detailed model (which simulates memory accesses and memory busy exactly) would run slower (more clock cycles for the same number of floating-point operations) than the higher-level model.

The data obtained from the individual sequence runs was summarized and used to compare against other forms of runs for the implicit code to validate the manual extrapolations that had to be made.

BTRI is broken into 4 different segments of code to facilitate simulation runs; these segments are identified as BTRI0, BTRI1, BTRI2, BTRI3. The segments are then combined into code sequences representing the pseudo-compiled lines which can be found in the listings of simulator input in appendix F. The simulation results for each segment are then combined to form a total operation count and total clock cycle count for those sequences in BTRI. These can be found in the table below where the first line for each segment represents the number of floating-point operations credited to that sequence. The second line for each segment gives the number of clock periods for that sequence. The floating-point result rate is given for each subsequence as an indication of how that particular form of code would perform on the FMP.

```
VECTOR LENGTHS          100      200      400      1600      1600      600
                                                64-MODE   32-MODE

4330-4815 BTRI0       11900    23800    47600    190400    190400    21400
CLKPD                   885     1561     2861     10661      5461     4161
          BTRI1       22500    45000    90000    360000    360000   135000
CLKPD                  1361     2401     4401     16401      8401     6401

TOTALOPS              34400    68800   137600    550400    550400   206400
TOTALCLK               2246     3962     7262     27062     13862    10562
4330-4815 FPRATE    0.957E9  1.085E9  1.184E9   1.271E9   2.481E9  1.221E9


4820-5476 BTRI2       30000    60000   120000    480000    480000   180000
CLKPD                  2041     3601     6601     24601     12601     9601
          BTRI0       11900    23800    47600    190400    190400    71400
CLKPD                   885     1561     2861     10661      5461     4161
          BTRI1       22500    45000    90000    360000    360000   135000
CLKPD                  1361     2401     4401     16401      8401     6401

TOTALOPS              64400   128800   257600   1030400   1030400   386400
TOTALCLKS              4287     7563    13863     51663     26463    20163
4820-5476 FPRATE    0.938E9  1.064E9  1.161E9   1.247E9   2.434E9  1.198E9


5490-6015 BTRI2       30000    60000   120000    480000    480000   180000
CLKPD                  2041     3601     6601     24601     12601     9601
          BTRI0       11900    23800    47600    190400    190400    71400
CLKPD                   885     1561     2861     10661      5461     4161

TOTALOPS              41900    83800   167600    670400    670400   251400
TOTALCLKS              2926     5162     9462     35262     18062    13762
5490-6015 FPRATE    0.894E9  1.015E9  1.107E9   1.188E9   2.320E9  1.142E9


6020-6100 BTRI3        5000    10000    20000     80000     80000    30000
CLKPD                   341      601     1101      4101      2101     1601
6020-6100 FPRATE    0.916E9  1.034E9  1.135E9   1.219E9   2.380E9  1.171E9
```

Note that lines 4820 through 5476 and lines 6020 through 6100 are executed IMAX times, where "IMAX" is the maximum length of the column vector being solved by the tridiagonal algorithm. Thus IMAX will become either JMAX, KMAX, or LMAX depending on the sweep direction. Summing the values for the two sets of subsequences:

```
4330-4815&5490-6015   76300   152600   305200   1220800   1220800   457800
CLKPD                  5172     9124    16724     62324     31924    24324

4820-5476&6020-6100   69400   138800   277600   1110400   1110400   416400
CLKPD                  4628     8164     4964     55764     28564    21764
```

To find appropriate values for the execution of the full BTRI, the values of IMAX that are reasonable must be determined. This is attacked by establishing characteristic mesh sizes for the entire implicit solution. Since the primary interest is in the sensitivity of the FMP to small vector sizes, parameters are chosen as follows:

VLENGTH	DIMENSIONS	SLAB SIZE	IMAX	BTRI VL
100	6*6*6	2	6	12
200	8*8*8	3	8	25
400	10*10*10	4	10	40
1600	16*16*16	6	16	100
57624	100*100*100	6	100	600

The purpose of this relationship chart is to pick the lengths of operations in BTRI depending on the vector lengths used in FILTRX, FILTRY, RHS, etc. Obviously, the mesh dimensions chosen for all but the last case are smaller than would be normally used. However, it is equally obvious that if the FMP can perform well on these small meshes, and even better on larger meshes, then the threshold of one gigaflop can be attained by a broad range of problem models.

The number of planes appearing in a 'slab', a function in the program of the variables JSL, KSL, LSL, is based on the memory space available. In the case of the first meshes given, it can be seen that the entire mesh could be held in Main Memory and thus no 'slabbing' would be necessary. In addition all transposes would be performed in Main Memory instead of between Main and Intermediate Memories, with a consequent increase in Map Unit performance. For the purpose of this report however, all operations are performed as if the mesh variables had to be retained in Intermediate Memory, regardless of the "typical" dimensions chosen for the flow variable arrays. Using the values of IMAX and vector lengths given above:

VLENGTH-IMAX	100-6	200-8	400-10	1600-16 64-MODE	1600-16 32-MODE	600-100
BTRI TOTAL OPS	492700	1263000	3081200	18987200	18987200	42097800
BTRI TOTAL CLKS	32940	74436	166364	954548	488948	2200724
BTRI GFLOPS	0.935	1.064	1.158	1.243	2.427	1.196

Thirty-two-bit simulation was limited to the set of runs at vector lengths of 1600. This seems to represent a reasonable length for some problem solutions, and was chosen as a midpoint in the studies of FMP performance done for this report.

To provide data for analysis of the behavior of the three-dimensional implicit code, simulation was performed on the following:

FILTRX/AMATRX	
BTRI	
FILTRY/AMATRX	
BTRI	
FILTRZ/AMATRX	
BTRI	
RHS	
----------	Subtotal for implicit without viscosity
VISMAT	
VISRHS	
----------	Subtotal for implicit with viscosity, laminar flow
MUTUR	
----------	Total for implicit

The pseudo-compilation of FILTRX, FILTRY, RHS, VISMAT, VISRHS, and MUTUR were translated into simulator input (see appendix F) and run for the vector lengths mentioned previously. As can be seen from the pseudo-compilation there are a set of map operations that must be accomplished at the beginning of each sweep loop. The results of these map operations are interlocked to subsequent arithmetic operations by use of the read and write keys. The remaining map operations are performed concurrently with arithmetic operations during the remaining passes through the loop. Since each FILTRX/AMATRX sweep is simulated only down to the BTRI call statement, the times given by the simulator assume that all map and vector operations must be completed before entry into BTRI. Since this is not the actual case the timings given are considered worst case, since in fact the last few vector and map operations can proceed concurrently with the initial functions appearing in BTRI.

The timings for FILTRX and FILTRZ are essentially the same, since they are governed by the "gather-record" operations. FILTRY requires "single element gather" operations however, and must be simulated separately.

All simulation of the implicit code assumes that the code and the flow variables have been staged to Intermediate Memory from either disk or backing storage, whilst a previous code is running on the FMP. Thus no explicit I/O is calculated into the analysis since all data transfers to the "outside world" are assumed to be overlapped with other program executions.

Thus, simulation results for a single run of BTRI for all desired vector lengths (described previously) must be multiplied by three. Likewise the FILTRX/AMATRX simulation results can be multiplied by two (to represent FILTRY/AMATRX as well); the remaining runs are then factored in on a one-for-one basis. The following matrix summarizes the results from the set of runs for the implicit code that can be found in appendix F. The multiplication factor used to form the subtotal for the left-hand side appears in the left-hand column of the chart. The values given for each code segment represent the single-run counts for that code.

The data that appears in the chart for FILTRX and FILTRY are taken directly from the simulation output and represent two iterations through the slab processing loop, in order to amortize the startup costs of the initial map operations. Since the chart represents only one pass through each sweep for one single slab, the multiplication factor must be reduced by one-half for each FILTRX, FILTRY, and FILTRZ. Since the data for FILTRX and FILTRZ are the same, the factor for that line is 0.5*2 (FILTRX + FILTRZ) = 1, while the factor for FILTRY is 0.5*1 = 0.5.

FACTOR	VLENGTHS	100	200	400	1600 64-MODE	1600 32-MODE	57624
3	BTRI OPS	492700	1263000	3081200	18897200	18897200	42097800
3	BTRI CLK	32940	74436	166364	954548	488948	2200724
FPRATE	BTRI	0.935E9	1.060E9	1.158E9	1.243E9	2.427E9	1.196E9
1	FILTRX	24200	48400	96800	396300	396300	13945006
1	CLKPD	3102	5128	9426	34536	17656	1268201
FPRATE	FILTRX	0.487E9	0.590E9	0.641E9	0.701E9	1.370E9	0.715E9
.5	FILTRY	24200	48400	96800	396300	396300	13945006
.5	CLKPD	10697	20021	39331	162271	81371	4532289
FPRATE	FILTRY	0.141E9	0.151E9	0.153E9	0.149E9	0.297E9	0.193E9
1	RHS OPS	21300	42600	85200	340800	340800	12273911
1	RHS CLK	1724	3051	5626	21076	12276	742385
FPRATE	RHS	0.772E9	0.872E9	0.946E9	1.011E9	1.735E9	1.033E9
SUBTOTAL	OPS	1535700	3904200	9474000	5762850	5762850	1.595E8
SUBTOTAL	CLKS	108994	241497	533809	3000391	1537461	10144296
FPRATE	W/O VISC	0.881E9	1.010E9	1.109E9	1.200E9	2.343E9	0.983E9
VISMAT-VISRHS		58800	117600	235200	940839	940839	33882908
CLKPD		4218	7442	13642	51028	25700	1787586
FPRATE	W/VISC.	0.871E9	0.987E9	1.078E9	1.152E9	2.288E9	1.185E9
SUBTOTAL	OPS	1594500	4021800	9709200	58567689	58567689	1.934E8
SUBTOTAL	CLKS	113212	248939	547451	3101359	1588091	10933510
FPRATE	W/VISC	0.880E9	1.010E9	1.108E9	1.180E9	2.304E9	1.106E9
MUTUR OPS		26050	52100	1.2645E5	4.993E5	4.993E5	1501100
MUTUR CLKS		7000	8140	12370	43464	2.25E4	1.529E6
FPRATE MUTUR		0.25E9	0.40E9	0.639E9	0.718E9	1.384E9	0.755E9
GRAND TOTAL OPS		1620550	4073900	9813400	58984489	58984489	2.084E8
GRAND TOTAL CLKS		120212	257079	560476	3094819	1586841	13374882
TOTAL FPRATE		0.843E9	0.990E9	1.094E9	1.191E9	2.323E9	0.974E9

As the chart shows, the minimal one-gigaflop rate is achieved at vector lengths of 400. The anomalous behavior of the long vector case is due to two effects. First, if one examines the FPRATES for FILTRX and FILTRY, the rate seems to drop off at the maximum vector length. This is due to the simplification used to run the codes. It turns out that the final map operation (line 800 in the FILTRX and FILTRY simulation input) dominates the Map Unit behavior. As was mentioned previously, the disengagement of the BTRI CALL from the FILTRX/AMATRX sequence for simulation purposes imposes an additional penalty at the end of the simulated sequence while the simulator is 'idling' down. This can be seen by executing the sequence with repeat factors of 1, 2, and 3, where the 'idling' will be amortized over the total execution.

The second effect is seen in BTRI. The vector lengths used in BTRI are actually only 600, since the iteration within BTRI is recursive in the sweep direction. Thus while all of the other data reflects increasing vector lengths from 100, 200, 400, 1600, and 60000, BTRI lengths are actually 100, 200, 400, 1600, and 600. This was an unfortunate choice of parameters, but was established early in the study to ease analytical evaluation of the implicit algorithm. The effect of the tapering of the "performance curve" which could be plotted from the above results is then due to two unnecessary and eliminatible elements. The first can be resolved by resimulating the entire mass with other parameters; the second can be resolved by moving the gather operation (at line 800) earlier in the code (a task for the compiler), and including the first sequence from BTRI as part of FILTRX, FILTRY, and FILTRZ. The FPRATE shown for vector lengths of 57624 is not as high as the original one-gigaflop objective. This was mostly due to the effect of the slow Intermediate Memory to Main Memory transpose operations required by the data just described, which led to a new series of runs with a slightly revised version of STEP.

In this version the Q, X, Y, and Z meshes are fully transposed into a form suitable for direct processing by FILTRY. This transposition is performed from Intermediate to Intermediate Memory during the BTRI computations in the FILTRX pass, and requires hand-coding of the gather operations immediately preceding the CALL to BTRI in FILTRX. Once this transpose is complete, FILTRX, FILTRY, and FILTRZ all execute in the same manner, and thus their rates will be identical.

To prove this was possible it was necessary to merge the simulation input files for BTRI and FILTRX, so that the code can be run as it would be executed on the FMP (with the shutdown times imposed by the simulation system being overlapped by following code).

Unfortunately this scheme generates too many simulator input cards for reasonable values of IMAX (in BTRI). Therefore, a set of runs were made with the revised FILTRX/AMATRX/ TRANSPOSE/BTRI aglutinized, and the following results were obtained:

```
VLENGTH=BTRIVL      400=40     1600=96    60000=600        1600=96  (32BIT)

IMAX=1  FLOPS       1.064E5    3.337E5    8.130E6          3.335E5
IMAX=1  CLKPD       1.08E4     2.83E4     7.29E5           1.578E4

IMAX=2  FLOPS       1.349E5    4.001E5    8.543E6          4.000E5
IMAX=2  CLKPD       1.36E4     3.29E4     7.51E5           1.877E4

IMAX=5  FLOPS       2.177E5    6.001E5    9.805E6          6.001E5
IMAX=5  CLKPD       2.17E4     4.68E4     8.16E5           2.775E4

IMAX=7  FLOPS       2.737E5    7.329E5   10.636E6          7.329E5
IMAX=7  CLKPD       2.72E4     5.60E4     8.60E5           3.373E4
```

Extrapolated data was then determined for IMAX values of 10, 16, and 100 using the following extrapolation equations:

$$\text{DELFLOPS} = \frac{\text{FLOPS(IMAX=7)} - \text{FLOPS(IMAX=1)}}{6}$$

DELRATE = The asymptotic FPRATE as a function of IMAX for the additional FLOPS (determined graphically to 3 significant digits).

$$\text{XFLOPS(IMAX} > 7) = \text{DELFLOPS}*(\text{IMAX} - 7)$$

$$\text{XCLKPDS(IMAX} > 7) = \frac{\text{XFLOPS(IMAX)}}{\text{16E-09*DELRATE}}$$

$$\text{FPRATE(IMAX} > 7) = \frac{\text{FLOPS(IMAX} = 7) + \text{XFLOPS}}{\text{16E-09*(CLKPDS(IMAX} = 7) + \text{XCLKPDS)}}$$

```
IMAX=10  FLOPS      3.592E5
IMAX=10  CLKPD      3.558E4
IMAX=10  FPRATE     0.631 GFLOP

IMAX=16  FLOPS                1.331E6                   1.331E6
IMAX=16  CLKPD                9.75E4                    6.064E4
IMAX=16  FPRATE               0.853 GFLOP               1.388E9

IMAX=100 FLOPS                           52.236E6
IMAX=100 CLKPD                            3.03E6
IMAX=100 FPRATE                           1.08 GFLOPS
```

where the pair of vector lengths shown represent the typical lengths in FILTRX, AMATRX, RHS, VISMAT, VISRHS, MUTUR, and appearing in BTRI, respectively. The purpose for using this range was to find the 'knee' of the curve of performance versus

ORIGINAL PAGE IS OF POOR QUALITY

mesh sizes. The range of IMAX was dictated by the maximum number of input statements manageable by the simulator. Thus, the values had to be extrapolated for the entire code block for IMAX values of 10, 16, and 100 which represent the mesh sizes given in the previous table. It was determined from the runs that for any value of IMAX, the K direction transpose is completely covered by the BTRI calculations. This is due to the fact that the map operations in BTRI are Main Memory to Main Memory, while the transpose operation requires only the Intermediate Map Unit.

The extrapolation was tested against the previously-run implicit data, which required hand-aggregated values for vector lengths of 57624, and was found to be within 1% of that data. Taking the values of FLOPS AND CLKPD for RHS, VISMAT-VISRHS, and MUTUR from the previous table for vector lengths of 400, 1600, and 60000 results in

VLENGTHS	400-10	1600-16	60000-100	1600-16 3 2-MODE
FILTRX FLOPS	3.592E5	1.331E6	5.223E6	1.331E6
FILTRX CLKPD	3.558E4	9.75E4	3.03E6	6.064E4
3*FILTRX FLOPS	1.078E6	3.993E6	1.567E8	3.99E6
3*FILTRX CLKPD	1.067E5	2.925E5	9.108E6	1.82E5
RHS FLOPS	85200	340800	1.227E7	3.41E5
RHS CLKPD	5626	21076	742385	1.23E4
STEP W/O VISC	1.163E6	4.334E6	1.69E8	4.33E6
STEP CLKPD	1.123E5	3.136E5	9.850E6	1.94E5
STEP FPRATE	0.647E9	0.864E9	1.072E9	1.395E9
VISMAT-VISRHS	235200	940839	33882908	9.41E5
CLKPD	13642	51028	1787586	2.57E4
MUTUR FLOPS	1.265E5	4.993E5	1.847E 7	4.99E5
MUTUR CLKPD	1.237E4	4.346E4	1.529E6	2.25E4
TOTAL FLOPS	1.525E6	5.774E6	2.214E8	5.771E6
TOTAL CLKPD	1.383E5	4.081E5	1.317E7	2.43E5
FINAL RATE	0.689E9	0.884E9	1.051E9	1.487E9

The counts shown above are for one pass through each sweep direction (or one full slab processed). Extending these numbers for all slabs and 256 time steps gives the following:

TOTAL RUN FLOPS	1.17E9	4.43E9	9.07E11	4.43E9
TOTAL RUN CLKPD	6.64E6	3.13E8	5.39E10	1.87E8

The final revision of the simulation input which yields the FLOP rate shown here is felt to be practicable in terms of both programmability and compilability. It takes into account the gather operations needed for collecting the data for each slab and includes three extra gather operations to represent the initial gather functions which must be accomplished before the first pass of the loop can be processed. If this sequence were to be extended to the proper number of passes, then the loop count would be 16, for 16 slabs needed in the 100x100x100 mesh case. In one experiment with the simulator the three initial map operations were removed and the FILTRX, FILTRY, and FILTRZ timings were reevaluated. The difference in execution time of the entire ensemble due to the additional map operations turns out to be less then 4%, which is less than the probable accuracy of the total estimates.

From the several attempts at recoding and simulating the implicit code, it appears that all transpose and gather operations could be overlapped completely with careful 'hand-coding' or 'extremely sophisticated compiling'. In this instance the execution rate could become as high as the 1.12 gigaflops achieved by the unadorned BTRI routine by itself.

An important concern in any of these code simulations is the degree to which the system is balanced. That is, are there any major components idle or nearly idle while others are at maximum utilization? Examination of the simulation results will disclose two quantities of interest beyond the FPRATE and CLKPD--VECBZ and MAPBZ which reflect the percentage utilization (degree unit is busy) of the Vector and Map Units, respectively. In the results for the implicit code it can be seen that the Map Unit is occupied for at least half the time, and the Vector Unit is over 90 percent busy. As long as the Map Unit is never busier then the Vector Unit, and the Vector Unit is busy between 93 and 98 percent of the time, one can be confident that consumers are getting their money's worth.

Another observation of interest is that the curves of performance versus vector length (or mesh size) yield some data that points to an asymptotic behavior of the GIGAFLOP curve. Unfortunately project time hasn't permitted investigation of the location of the exact knee of the performance curve. It would seem that a simple graphing of the points for lengths of 400 (dimension 10x10x10), 1600 (dimension 16x16x16), and 60000 (dimension 100x100x100) will show that at vector lengths of 6400 (dimension 24x24x24) and greater, the FLOP rate of the proposed FMP will hover around 1.000 gigaflop. The implication is obvious that for the range of reasonable mesh sizes -- 30x30x30 through 100x100x100 -- the performance curve is relatively flat and close to one gigaflop (plus or minus 10%). This range is achieved even when all arrays are kept in Intermediate Memory throughout the computation!

As mesh sizes grow too large to be held entirely in Intermediate Memory they can be held in the Backing Store and swapped into Intermediate Memory, then transposed by the Intermediate Map Unit. Although this has not been simulated, the excess capability available in the current map operations points to the likelihood that meshes on the order of 200x200x200 up to 240x240x240 could be processed at close to the one-gigaflop rate. It should be obvious, however, that the total problem solution time for the largest case will be greater by far than the acceptable threshold of 10-15 minutes CPU time for the nominal 100x100x100 problem.

The reality of the above conclusions rests on

1) the degree to which the simulation systems representing the CDC FMP are true and valid engineering models of the actual hardware design;

2) the sufficiency of the object code produced for simulation;

3) the validity of the extrapolations made based on the simulation results.

It is expected that analysts at NASA and CDC will continue to exercise these simulators for their own research to determine the quality of the aforementioned issues, and perhaps to locate more accurately the knee of the performance curve.

5.2 THREE-DIMENSIONAL EXPLICIT CODE

As stated previously, the explicit code was considered in a different light than the implicit code. In particular, the vectorizations attempted were limited to those available on the STAR-100 computer. This permitted measuring the ability of programmer, compiler, and machine architecture while assuring that the metric conversions yielded the same answers as the original code. The characteristics of the explicit code of interest to this study have been —

1) the implicit sections (LZI, LYI) which in mathematical form are similar to the implicit portions of the implicit code, but are constrained to smaller vector lengths;

2) the explicit solver schemes (LX, LY, and LZ) which limit the potential vector lengths because of J and L recursion;

3) the method of characteristics (CHARAC) which exhibits conditional processing of data, based on the data.

As in the case of the implicit code, the routines of interest were vectorized, then actually compiled (not pseudo-compiled as in the implicit code) using the STAR-100 compiler. The STAR object code was then transliterated directly into FMP object code and the results were simulated. The simulation involved fragmenting the sequences into smaller portions for processing by the detailed level simulator (LVL2), then aggregating them into whole strings for simulation at the higher (LVL1) level. The first attempt at this led to FPRATES which were 100-300 megaflops for the 30x30x30 case. This resulted from the transliteration yielding very few dyadic and triadic operations in the pipelines. The simulation input was then subjected to a "sophisticated compiler" consisting of an experienced FMP programmer. The resulting input was resubmitted to the high-level simulator and the following results were obtained. (See appendix F for the simulator runs and FMP FORTRAN source code.)

The routines VLX and VLYI were run through the high-level simulator (LVL1) for a variety of vector lengths to demonstrate the performance range for those sequences. Runs for vector lengths above 100 were difficult to do for VLYI because of the overflow of the simulator instruction buffer. From this raw data a table of total performance is developed later.

VLENGTH		30	100	600	1000
VLX	FLOPS	8833	29677	119670	
	CLKPD	1129	2398	7672	28822
	FPRATE	0.489E9	0.773E9	0.974E9	1.040E9
VLYI	FLOPS	1.31E6	3.04E6	4.57E7	1.17E8
	CLKPD	1.09E5	2.57E5	3.26E6	8.82E6
	FPRATE	0.750E9	0.738E9	0.824E9	0.826E9

The VLX rates given for vector lengths of 30 and 100 correspond to the mesh sizes 30x30x30 and 100x100x100, respectively. The FLOP and clock period counts for the corresponding VLYI simulation runs reflect the processing of planes of data in the J by K planes only. The repeat counters at lines 840, 1680, and 1900 of the VLX simulation input had to be scaled by one-fifth to make the simulation fit within the GPSS code limitation. The effect of this scaling is to reduce the effective FPRATE by 0.01% from what should be the actual rate. In order to provide correct values of FLOPS and CLKPD for the 100x100x100 case, the quantities must be increased as follows: lines 840 through 950 yield a total of 182000 FLOPS for the VLENGTH=100 case in the simulation results. This number should be five times greater, due to the scaling used, or 910000 FLOPS. The total clock periods required for a 5X extension of these loops is 1.42E5 for a revised FLOP rate of 0.73E9.

The raw data above was further extended by the factors 30 and 100 respectively to reflect the number of operations needed for all three dimensions for that particular mesh. TURBDA, LYC, and CHARAC were estimated by hand. First, CHARAC could not be estimated with the simulator since the number of operations credited by the simulator is 100% rather than the number of one-bits in the control vector. Second, TURBDA and LYC are obvious vectorizable entities whose performance can be estimated easily and directly from the code.

```
                    EXPLICIT CODE SUMMARIES
MESH SIZES          30X30X30            100X100X100
TURBDA
   FLOPS            4.87E5              1.81E7
   CLKPD            4.05E4              1.46E6
   FPRATE           0.75E9              0.775E9

LYC
   FLOPS            9.98E5              3.07E7
   CLKPD            1.40E5              2.54E6
   FPRATE           0.44E9              0.75E9

LYI
   FLOPS            3.93E7              3.13E8
   CLKPD            3.27E6              1.93E7
   FPRATE           0.750E9             1.015E9

LX
   FLOPS            7.98E6              2.97E8
   CLKPD            1.01E6              2.40E7
   FPRATE           0.489E9             0.773E9
```

Extrapolation of full explicit performance is fraught with problems due to the lack of accounting for overlap of operations from the end of one routine into the beginning of another routine, or on the contrary, the conflicts that delay the start of a routine due to a previous routine. Time and resources didn't permit a complete evaluation via simulation. Instead, the simplification used for extrapolation was to assume that LY, LX, and LZ all execute about the same number of cycles for the same number of operations on a symmetrical mesh. The same simplification was used for LYC, LZC and for LYI, LZI. Grand total then consists of a linear combination of LX, TURBDA, LYC, and LYI routines as follows:

6*LX
2*TURBDA
4*LYC
4*LYI

```
GRAND TOTALS
FLOPS               2.1E8              3.19E9
CLKPD               1.97E7             2.34E8
FPRATE              0.664E9            0.85E9
TOTAL RUN FLOPS     5.38E10            8.17E11
TOTAL RUN CLKPD     5.06E9             5.99E10
```

5.3 SPECTRAL WEATHER MODEL

The FFT routine and Legendre (SPCFOR) routines were simulated with the following mesh parameters

FFT length	32	0.518 GFLOP
FFT length	100	0.858 GFLOP
FFT length	200	0.973 GFLOP
FFT length	2000	1.145 GFLOPS

FFT 32-bit	100	1.326 GFLOPS
FFT 32-bit	500	2.027 GFLOPS
FFT 32-bit	2000	2.246 GFLOPS

Legendre transform -- 25 layers, 6 waves, 15 gaussian latitudes

0.837 GFLOP

Spectral model overall estimate in its present configuration

0.879 GFLOP

The spectral code was extrapolated by assuming:

> The FFT and SPCFOR/FORSPC routines constitute the bulk of the processing;
>
> The remaining code sequences are vectorizable to the same degree as the FFT and SPCFOR routines;
>
> FORSPC data can be estimated from the simulated data for SPCFOR.

A formula from reference 3 was then used to estimate the total number of FLOPS. Two problem sizes were used to establish the performance range. The first -- 25 layers, 6 waves, and 15 gaussian latitudes -- was the scaling in the original spectral metric provided by Ames. The second size is one determined by CDC and the code originators to be reasonable for actual production runs of this type research model--48 layers, 21 waves, and 55 gaussian latitudes. The following table summarizes the data and extrapolation.

```
                         PROBLEM     SIZE

                       25,6,15          48,21,55

TOTAL FLOPS            3.22E6           8.65E7

FFT
FPRATE                 0.970E9          1.11E9
% OF TOTAL FLOPS       0.67             0.41
FLOPS                  2.16E6           3.55E7
CLKPDS                 1.39E5           2.00E6

SPCFOR
FPRATE                 0.610E9          1.14E9
% OF TOTAL FLOPS       0.14             0.34
FLOPS                  4.51E5           2.94E7
CLKPDS                 4.62E4           1.612E6

FORSPC
FPRATE(ESTIMATE)       0.5              0.9
% OF TOTAL FLOPS       0.06             0.17
FLOPS                  1.93E5           1.47E7
CLKPDS                 2.42E4           1.0E6

TIME WEIGHTED SUM OF FFT,  SPCFOR,  AND FORSPC
% OF TOTAL FLOPS       0.86             0.92
FLOPS                  2.77E6           7.96E7
CLKPDS                 2.09E5           4.61E6
FPRATE                 0.827E9          1.08E9
```

5.4 FINITE DIFFERENCE WEATHER MODEL

Only one routine from the GISS code was simulated. LINKHO had become the major concern of CDC analysts when original projections showed performance below 50 megaflops for that routine. The size of the GISS code had prohibited a complete analysis to the recoding, compiling, and simulation level. As a result, LINKHO and AVRX were studied to see if there were some fundamental problems that existed in the code which reflected serious problems in the FMP architecture. The LINKHO subroutine was analyzed and the first computational section was chosen for simulation (due to the sheer volume of code in LINKHO) and a single page, felt representative of the majority of the routine, was subjected to the simulator. The simulation input data can be found in appendix F.

The AVRX routine was estimated by hand, with the inclusion of the effect of vector startup times. The extrapolation used is based on the code as presented in figures 7 and 8 of Division 4, the analysis of the weather codes. There are 4*23*5*2+6*4*2=968 floating point operations reflected in figure 7. There are (23*5*2+6*2)*2=484 floating point operations reflected in figure 8. Assuming the compiler is capable of scheduling simultaneous scalar execution with the Vector Unit for some computations, (13+6)*2*2=76 machine cycles would be required to complete the sequence in figure 7, for a rate of 0.8 gigaflop. The sequence in figure 8 would require (3+6)*2*2*2=72 machine cycles, for a rate of only 0.42 gigaflop due to the inherently short vector lengths. Together, a total of 1452 floating point operations, requiring 148 machine cycles, yields a rate of 0.62 gigaflop. Note that these timings are for 64-bit mode, while 32-bit mode is sufficient for the mathematics in the AVRX code.

The rate given is for a relatively course grid; using the resolution present in the original—Ames-supplied metric results in a grid of 46*72. This denser grid should require 24*43*5*2+624*2=10608 floating point operations based on the sequence in figure 7 and (43*5*2+6*2)*2=884 operations in the sequence in figure 8. The length of vectors in this fine grid model in figure 7 is 1104 elements, and that in figure 8 is 43 elements.

With the longer vector lengths it is expected that the Scalar Unit will sustain parallel execution with the Vector Units. This last example would require (138+6)*2*2=576 machine cycles for the sequence in figure 7 and (6+6)*2*2*2=96 cycles for the sequence in figure 8. Thus, if the greater resolution given in the original model is permitted, the FMP could achieve

$$(10608+884)/(576+96)=1.07 \text{ gigaflops}$$

for this portion of the GISS code.

5.5 REFLECTIONS ON THE PERFORMANCE ANALYSIS

The process of analyzing the performance of a candidate computer system for this report has become more involved than originally anticipated. This can be attributed to two factors:

1. The simulator system could not process a total program; programs had to be broken into pieces, and each piece run independently. Some runs had to be made with the overlapping code between pieces to determine if the independent runs are valid. (The effect of splitting code is to create 'end-case' values for some cases, as the simulator counts all cycles needed to complete the last operations. In a non-split case this last operation may be overlapped with operations from a subsequent piece of code. The question then becomes how to compute the aggregate total of clock cycles in these 'end-case' situations.) In any event the composite timings must be computed by hand, with the attendant risk of manual error. The simulator should be modified to accept a much larger sequence of instructions, both by eliminating the expansion of input cards that now takes place when an R card is encountered and by enlarging the instruction buffer itself. Using the CYBER 175 revealed that the computer time required for the high-level (LVL1) simulator to process large programs is quite within reason (1 to 2 minutes for the entire implicit code perhaps). In addition, the FPRATE, FLOPS, and CLKPD values must be output in some machine-readable form so that further extension or computation can be performed with a FORTRAN or BASIC program to produce final outputs for analaysis.

2. Until the full data for a given code had been aggregated and analyzed, weak points in the hardware complex, or the coding or compiling scheme, could not be evaluated properly. The result of this was that after laborious simulation, aggregation, and projection, an area of the software/hardware was found to be eligible for more 'tuning'. Once the code was modified the same process was repeated again, and again, and again. In addition to the LVL1 runs, segments had to be submitted to the LVL2 simulator to make sure that the memory conflicts and scalar interactions of the detailed simulator produced consistent performance figures at both levels of simulation. Computer time did not prove to be the limiting factor in this process; instead the resources able to perform the analysis and system design modification became the scarce commodity on this program.

If time had permitted, the range of problem sizes should have been expanded to encompass the key areas of the performance curve (the area of the knee) so that the optimal problem sizing could be determined. It was felt that establishing the range was a more important objective than proving the performance for a given problem statement. (Hence no attempt was made to make complete runs for the 100x200x50 grid sizes.) The symmetrical grids of 30x30x30 and 100x100x100 were used as end-points because they represented two worst-case situations -- one with extremely short vector lengths in all dimensions, the other with maximum data storage and mapping operations in all directions. In the CDC FMP, if all problems could be restricted to forms like 100x200x50, the flow mesh would be stored and processed differently, with worst-case map operations being performed along the short (50 element) axis only.

For future consideration, any data dependent features of metrics should be identified and some method of parameterization of the simulation and performance analysis developed. For example, in the CHARAC routine extrapolation, a simple-minded scheme for determining how many operations to CREDIT to the simulation was introduced.

Certainly a better set of criteria could be developed for later simulation runs. Similar parameters could be provided for routines such as MUTUR in the implicit code.

Some ambiguity exists in the manner in which FLOPS are counted. Some analysis will credit

```
A=-1

A=ABS(B)
```

as one floating point operation each; this has not been done in this report. Other vendors claim that

```
B=1/C
```

is one floating point operation while

```
B=A/C
```

is counted as two operations. More complicated to analyze is the presence of a hardware function such as SQRT. How many floating point operations should be credited to this operation? Some analysts claim one, others claim 5, and some even claim 13. In the codes simulated for this report, the SQRT approximation took 11 vector operations (where the divide is counted as one operation). It should thus be claimed that SQRT be counted as 11 operations if a hardware SQRT were to be implemented. A standard set of criteria should be developed for this aspect of performance analysis for any future studies.

If a variety of simulators is to be used to evaluate a variety of candidate architectures, it would be desirable for NASA-Ames to develop a simple 'validation' routine which could be used to provide a measure of total FLOPS, FPRATE, and CLKPD (clock periods) for a variety of functions whose totals could be computed reliably and canonically. This would provide a form of 'performance diagnostic' to verify certain characteristics of the simulation and extrapolation system being used. It is only after confidence can be established in the modeling system that the results of analyses such as these can be viewed realistically.

5.6 BOTTOM LINE

A few general comments should be made about the results just presented.

1. The implicit and explicit codes run in about the same amount of time. For the 64-bit version, the implicit code would require 14.4 minutes for 256 time steps while the explicit version requires 15.9 minutes for 256 time steps.

2. The explicit code runs substantially less than the one-gigaflop rate in the form simulated for this study. It was discovered that if a hardware square root were employed, the explicit code FPRATE could be improved to 0.93 gigaflop, as long as 11 FLOPS could be credited to that operation. If the explicit code were to be restructured to process slabs similar to those in the implicit code, yielding minimal vector lengths (excluding CHARAC) of 600, the overall rate for the explicit code can be elevated to 1.004 gigaflops (even without the hardware square root). This is due to the fact that the LX, LY, LZ, and TRIDIA vector lengths of 100 are below the critical point in the FMP performance curve, while lengths of 600 are at the 'knee' of that curve.

3. A wide range of problem sizes can be accommodated by the FMP with performance rates at or near the one-gigaflop threshold, for the implicit code.

4. The minimal rate for any of the 'small problems', regardless of whether the 'aero' codes or weather codes are being dealt with, is greater than 600 megaflops. This is for mesh sizes of less than those expected to be employed on the actual FMP for production work.

5. The user of 32-bit forms for the codes can yield substantial benefits, not only a factor of almost 2 in performance, but in the ability to store larger problems in the available memory.

6. With some effort, even essentially 'non-vector' code forms such as MUTUR, CHARAC, and LINKHO can be structured to provide reasonably effective 'parallel' operations. The 'physics' solutions in this arena have not been attacked as yet, however.

7. The effective use of the hardware system involves a process of 'rethinking', 'restructuring', and code 'tuning' to achieve optimal results. The burden of achieving maximum performance with the current generation of technology must be shared by the software and hardware developers alike.

8. A reliable, approachable, valid, and credible computerized simulation system is essential to the successful evaluation, as well as implementation, of candidate architecture for the NASF.

6.0 SYSTEM DESIGN

One of the few unchanging proposals by Control Data for the NASF is the overall system design. It is based on an interconnection scheme called the Loosely Coupled Network (LCN) which was described in detail in reference 2. A more recent discussion of the LCN and the hardware and software support using the Programmable Device Controller (PDC) may be found in Division 4, Volume II of this report. At the outset it was realized that the NASF would realistically involve an amalgam of dissimilar computing equipment, including equipment alien to Control Data Corporation. The fundamental underpinning of the LCN is its ability to interconnect a wide variety of equipments spread over a vast geography. It is this property that makes the LCN immediately attractive for the NASF. Figures 20 and 21, shown in a previous section, illustrate the overall system organization with the FMP represented as a single functional entity. The front-end processors, or Support Processing System (SPS), are attached to the network trunk and thus share a number of the global resources (such as the 819 disks) with the FMP. A certain number of peripherals are attached directly to each SPS as a locally managed resource. The LCN makes use of one or more serial data trunks which, using existing products, can transfer data at peak rates of 50 megabits per second.

The serial trunk can accommodate up to 32 attachments so that all entities on the trunk become members of a "party line" and can thus communicate with complete flexibility, with the possibility that trunk contention may reduce effective bandwidth. To solve this problem the LCN has a unique contention resolution system which is discussed in Appendix A, Division 4, Volume II. Aside from the archival storage and graphics subsystems which are attached, the key system resource is the phalanx of high performance disks which are attached to the trunk and serve as the medium of staging jobs to/from the FMP from/to the SPS.

Specifications for the systems components in this design can be found in Volume II. The choice of SPS is highly dependent upon the amount of activity that is anticipated from interactive operation and upon the decision as to where the functions of mesh generation and compiling should be performed. For this report both functions are assumed to be done on the SPS. The sizing of the SPS at this point seems to require machines of the CYBER 175 class. Using standard software components the configuration of two such machines sharing an ECS memory for coordination and residence of system wide tables is called for. Perhaps later generations of hardware may be configured differently but, for purposes of costing and sizing the installation, it is expected that this level of SPS is necessary.

As there is much redundancy planned for the FMP, so too is the system designed around redundant trunks, processors, and disks so that it can survive one or more failures without losing function, or in most cases, without significant degradations in performance. The trunk capacity has deliberately been designed for excess performance. The major reason is the expectation that future peripheral and SPS technology will provide hardware capable of taxing the system.

Job flow has been discussed previously, however, the relationship of the redundant components was not considered. It is imperative that all components in such a complex system continually undergo strenuous exercise to make sure they are still viable. One important way to accomplish this is to rotate each redundant component into operational use on a regular basis, or better still, to have redundant components share the workload on a continuous basis. This latter system is employed in the Control Data NASF design. There are no "stand-by" components; instead, all elements of the system work on the problem at hand, with the option to shut down or be shut down and have the load automatically assumed by a partner. Thus all trunks will carry data over the execution of a job. When one trunk encounters a failure it goes off-line; without changing status tables, data transfers continue using the remaining trunks. The system can withstand the loss of an entire trunk and still maintain its throughput. In the case of an SPS going out of action, some loss in throughput may be sensed, but with proper sizing the loss of ability will be nearer 10% than the apparent 50%.

The proposed configuration is certainly open to many alternative strategies, with the provision that everything must adapt to, and be harmonious with the LCN. Several smaller processors may be indicated, rather than a pair of SPS processors. It is important to view the blocks in the diagram more as functions than as actual hardware components. For example, the FMP is provided with a CYBER 18 or similar class computer whose existence is dedicated to serving as the Maintenance Control Unit for the FMP. If the ultimate NASF consisted only of an FMP provided from Conrol Data, with all other equipment being alien, then the CYBER 18 would be considered integral to the FMP design. However, the fact has been stressed that all maintenance functions are communicated to the FMP via standard LCN messages. Thus software that can be interfaced to such messages can reside anywhere in the system. In fact, even with a CYBER 18 stand-alone MCU present, certain functions such as deadstart and recovery must also be provided on the SPS as a backup.

In a similar manner, the function of FMP manager can reside on the SPS or be distributed among a number of processors. As an example, the FMP manager function, or critical portions thereof, like the job start and stop and memory allocator, might be provided in backup software form on the CYBER 18 MCU. Then in the event of a catastrophic failure of the SPS or perhaps when the SPS is undergoing some special off-line tests, jobs could still be pushed through the FMP.

6.1 SYSTEM TRAFFIC FLOW

To determine the efficacy of the system design in concert with the FMP, a simulation system was created to assist in the analysis of job flow through the entire ensemble. Again, as for the FMP, it was found to be advantageous to create two levels of simulation one highly detailed in areas of design concern, the other intended to demonstrate the overall performance of the system. The simulators have been used to verify the sizing of the system components as well as the probable flow of control message and data traffic throughout the system due to workloads projected Ames study personnel.

Divisions 1 and 2 of Volume IV contain descriptions of these simulators and reference information in the form of a "user's manual". As with the FMP, the more detailed model was used to validate certain assumptions used in the higher level model. Among the design conclusions drawn from this model were early decisions regarding the size of data buffers needed at each PDC node in the system (1536 words or three disk sectors), and the effective transfer rates of the major peripherals, with details such as latency, message turnaround and trunk contention taken into account. Effective transfer rates for the 819 class disk to the SPS were found to be 7.67 megabits per second, for the graphics subsystem about 1.94 megabits per second, at worst, and the effective streaming rate to the FMP was found to be 17.4 megabits per second per disk controller.

These numbers were used to construct the higher level model which was then subjected to the workload analysis provided by NASA-Ames (NASF Usage Model, Version 79.001). A discussion of the results of this simulation run will be found in Division 11, of this report. The bottom line of this analysis is that while the network trunk is underutilized, the SPS and the FMP are involved in a tight race to see which one will be the bottleneck in the system. This is a good sign because it indicates that the key resources are in balance in the system. A second, and obvious observation from the simulation results is that a full 20 hours of actual FMP work cannot be done on the FMP during a continuous 20 hour interval due to the variety of activities needed to start such a session into operation and to wind it down.

Another obvious observation is that the tradeoffs between turnaround and throughput are clearly evident in the workload simulations. As long as a queue can be maintained at the FMP, the FMP can be kept busy and throughput maximized. The very existence of such a queue precludes the FMP being able to produce fast turnaround in any given random case. In fact, with all the SPS and trunk activity requirements in this system, the overhead imposed on turnaround amounts to about 50% of the job execution or more. Thus a 20-minute, large scale production run may require 10-15 minutes of additional processing, measured from startup in the system to emergence of the results at a user output device.

As noted in the writeup, the turnaround conclusions were reached without employing any form of priority algorithm, although the simulator is capable of handling priorities. It is expected that as the behavior of the system becomes understood, RADL and Ames will employ some of the other options and different workload schemes to test the system in more realistic and comprehensive ways.

6.2 SYSTEM SOFTWARE

The resource commitments involved in even moderate software development projects preclude the intrusion of elegance or a NIH (Not Invented Here) attitude in the NASF system. This is particularly true in the area of system software, which includes all software elements not residing on the FMP except the FMP compiler, loader, and FMP manager (which will reside on the SPS).

A hardheaded approach must be pursued in the specification and development of the system software because of the magnitude of such efforts and the far reaching impacts on the NASF of software maintenance, compatibility, training, and effectivity. First, the identification and specification of any system's software function that is not already in existence and proven in at least one operational environment must be avoided like "the plague". Given the scope of FMP oriented systems and applications software that must be developed in the short period of this development, the project can ill-afford to experiment with "just one more good idea". Therefore it is strongly urged that:

1) functional requirements for the system software include no more, nor less, than those functions already demonstrated and available in standard computer software offerings;

2) the system software not be "based" on an existing system, but instead be limited to that system and its standard derivatives as provided by the system's manufacturer;

3) in particular, the CDC NOS (Network Operating System) which manages and controls a wide variety of large scale and super computers (the CYBER 170 and CYBER 200 families) provides a good model to be used to specify functional and performance criteria for the NASF.

The choice of NOS comes about because of the considerable number of machine-years that have been engaged in its development and maturity as a large scale operating system. The high throughput in interactive mode makes it an excellent choice for SPS functions which must manage and direct the myriad of terminals which will ultimately be attached to the NASF. The NOS system will provide a standard software "package" which will include PDC and LCN support software for interconnection to a multiplicity of attached processors.

If the NASF is severely limited to using an extant system, then many applications can be transferred directly from CYBER family systems to the NASF for execution. Users thus need to become familiar with only one set of functions and command language constructs. Debugging of NASF software (with the exception of actual FMP debugging) can take place in any system supporting NOS. The FMP manager then becomes just another job to be scheduled for execution under the NOS system, and no special interfaces need be written for the FMP and thus imbedded in the basic operating system. Instead the software provides a standard communications methodology for the FMP manager and the FMP.

The specification of systems software functionality can then be drawn from the current Control Data NOS documentation with the following additions that are under development for release well in advance of the NASF availability:

1. Loosely Coupled Network system, permitting connection of a multiplicity of devices and processors.
2. Common data base manager which controls a common pool of shared mass storage devices on the network trunk.
3. A graphics support system similar in function to the standard SCOPE offering called GODAS.
4. An archival storage management system based on the Control Data MSS (tape library) concept.

Because of its conceptual position in the NASF structure, the heart of the system will be the SPS. Therefore, if more than one vendor is to be identified to supply hardware for the system, it is urged that the SPS vendor be held responsible for integration of all alien hardware, and adapting the alien hardware "drivers" (or local operating systems) into the system's software structure provided by the SPS.

6.3 SYSTEM AVAILABILITY

Division 6 of this report provides an update of the analysis of Reliability, Availability, and Mantainability factors that are expected to influence the operational use of the FMP. Some of this discussion has had further elaboration in the hardware design portion of this division of the report. There are three probabilities that are of interest to the consumer of such a large system.

1. The probability of a single failure occurring anywhere in the system during operational use time.
2. The probability of a failure occurring which causes an interruption in system service.
3. The probability that an undetected error will occur which will affect result data in a significant way.

As will be seen from the supplementary discussions on this subject, the first item is directly related to the total parts and interconnection count of the hardware. Since system failure can occur in software, some factor must be generated for that aspect which can be combined with the hardware factors. If a standard operating system such as NOS is used for extrapolation of the software error rate, it can be seen that from 100-200 hours of production time elapse between software errors, compared with the 9-20 hours for the hardware. Thus the hardware failure rate dominates in this arena.

The second probability is related directly to the first probability but includes the effects of redundant hardware components and other error controls such as SECDED. From the RAM study it can be seen that without SECDED the FMP is totally infeasible, for the MTBF is far too short to ensure a high system availability. Software and firmware error recovery and restart are essential also to prevent an unacceptable period of machine interruption. A system interruption is one that takes all the resources out of action for any job submitted. It is permissible then to abort a particular job because of the loss of a single disk, or an unrecoverable error in data trunks, and immediately start another job without calling the event a "system interruption".

This second probability is also highly dependent on the maintenance strategies employed, also discussed in the hardware design section of this report.

The third probability is the most difficult to measure, and is dependent on how much checking can be done in the total system. In the FMP, SECDED and the variably redundant Vector Units are attempts at providing checking in crucial areas. It cannot be expected that the pair of SPS machines could be checking each other, since they will both be kept quite busy managing the system. It is, however, worthy of consideration to postulate a more powerful pair of SPS processors which would each be able to perform redundant checking of the partner machine on a variable load basis (much as in the FMP Vector Unit). This has not been pursued further because of the implication of insideous change in the kernels of the otherwise standard software.

7.0 FACILITIES STUDY

In previous reports (refs. 1, 2) the various aspects of the NASF design and construction were addressed in some detail. For this report an update of the schedule and cost information are provided. The scheduling methodology is PERT based and contains certain data relating to lead times of proprietary Control Data technologies. For this reason the PERT schedule and cost update information will be supplied under separate cover (as Volume V) for limited distribution as an adjunct to the body of this report.

The basic principles of the schedule still hold true from the previous studies; however, some adjustments to times have been made as continued investigation mandated.

a) A 30-month detailed design and simulation effort is needed before final commitments can be made on costs and reliability.

b) Approximately 19 months after design completion, the NASF should be available for limited production work. Four months later the entire NASF should be in full-time production.

c) Phased installation of NASF components is recommended and projected. A first installment of one SPS, a set of 819 disks, at least one archival storage, one LCN trunk, and one graphics support system should be installed at Ames for software development during the FMP design phase.

d) The FMP installation would also be phased, with one-half the Intermediate Memory and no Backing Store being in the first increment. The Backing Store would be added in two increments with the final increment of the Intermediate Memory being scheduled as needed. This permits spreading out costs and resources over several years while still having production capacity on-line as early as possible.

e) Software development is predicated on the strategies outlined previously, that is, the only developments being the compiler, loader, and FMP manager, all of which can be developed on the SPS. This effort must start at the same time as the FMP design.

The implication of these latter points is that the parallel procurement strategy being contemplated by NASA, which proceeds to the point where detailed FMP designs are pursued and a single one chosen for construction, must consider the lead time for software development; this must begin concurrently with the design phase. This means that not only machine design must be evaluated and conducted during the parallel phase, but that software development will be going on also. Who pays for this effort with the attendant risks that the resulting effort will be discarded is a subject that needs to be fully explored before any further procurement action is undertaken.

7.1 RISK ANALYSIS

As evidenced by the fairly conservative design approaches taken on the FMP during this phase and the restriction of software development for the NASF, Control Data has labored to reduce the risks of this project to a level acceptable to rational customer and manufacturer management. The only technological risk presently permitted has been in the Backing Store area where a component is not yet available, but for which two backup schemes are viable -- use of a less aggressive memory part that is now coming into production, with a consequent reduction in memory capacity or reliability; delay of delivery of the Backing Store until a mature part is available, since the Backing Store can be absent in the initial configuration without severe penalties to performance.

Production of the CYBER 200 family using the memory and logic technologies planned for the FMP has now given real data points for production costs and lead times for the central processor. Assuming that the Control Data schedule and cost objectives become the project goals, then it can be stated unequivocally that the risks would be below those for a new machine product line development. This is primarily due to the fact that 90% of the technical effort will be based on existing systems, software and manufacturing techniques. The compiler, loader, and FMP manager development are the primary software risks, and the costs and schedules for these can be conservatively rated at par with risks undertaken on any Control Data project.

Note that this risk assessment is substantially more "upbeat" than past analyses which have reflected some degree of overly conservative "gloom and doom". The major reason is that the number one emphasis of this project phase has been to create an FMP and NASF that can be built with great confidence of meeting the cost, performance, and schedule objectives.

7.2 LOGISTICS SUPPORT

The operation of a large complex of equipment that necessarily characterizes the NASF engages many disciplines and involves many cost factors that can easily be missed. Assistance was obtained for this report from the CYBERNET DATA CENTER specialists who have experience with large systems. In particular, advice was sought from the STAR data center managers who provide "computational engine" services on STAR via a set of CYBER 170 front-ends, much like the FMP will be made available to the NASF user. The resulting paper (Division 9) describes the major considerations and probable resources required for long term operation of an NASF-scale system.

The figures given in this report and in the report on maintenance strategy should be considered conservative in light of the fact that the availability and mantainability of the NASF is expected to be much greater than existing systems. Factors included for systems upkeep, therefore, may well be reduced in the final analysis to 50% or less of the projected figures. It would be well to expect that at the outset, and for the first 24 months of NASF operation, the consevative approach to systems operation and support should be adhered to until enough experience is accumulated to guide cost reductions which are sure to lurk in the projections offered here.

7.3 PHYSICAL REQUIREMENTS

As part of the initial study (ref. 1) Control Data was asked to determine, to as great a level of detail as possible, the physical requirements for the actual installation site. The result of that effort was the prototype planning of an actual site containing the hardware projected for the NASF as considered in this study. Most of the equipments that absorbed the bulk of that installation have remained unchanged in quantity and configuration since that initial effort. The FMP has undergone the greatest change due to a reorganization of the memory systems, reduction of pipeline hardware and, quite signficantly, due to a change in the packaging system for the FMP Main Memory and logic. Figures 3.1-2 and 3.1-3 of the functional specification (Division 1, Volume II) show the revised floor plan of the resulting FMP.

The overall power and cooling requirements are quite similar to those reported previously (ref. 1), and the floor plan for the proposed installation has been left unmodified, since the revised NASF will fit into essentially the same area originally reserved for the NASF. The FMP power requirements differ somewhat from that proposed in the early study but the aggregate power for the installation is affected very little. Division 10 provides some detail and a summary of space, power, and cooling requirements for the proposed NASF.

APPENDIX A

CLOCK RATE AS A MEASURE OF COMPUTER PERFORMANCE

The rapid tick-tick-tick of a wrist watch is certainly different from the slow TOCK --- TOCK of a grandfather clock. That is, the two different timepieces have different clock rates, yet when properly tuned to their respective rates, they both have the same performance --- keeping accurate time, one revolution of the minute hand per hour. The performance of a timepiece cannot be judged, however, on the basis of its rate alone. It is necessary to have some other information such as what is inside the timepiece (to know what is accomplished per tick or per TOCK), or what its output is (is its performance - timekeeping - accurate). Without this additional information, a rapid rate cannot be judged good or bad since it may mean simply that the timepiece is not properly adjusted and therefore is not keeping accurate time.

In order to compare one computer with another, an oft-used measure is to compare the clock rates of the computers. The assumption is that if one computer has a clock rate, say twice as fast as another, then it will produce twice the result rate as the comparison machine. In fact, however, the performance of a computer should be measured in the number of basic functions accomplished in some unit time. To derive this by only counting clock periods/unit time assumes that basic functions/clock period is constant. It does not take much thought to show that this is not constant, but rather a function of machine architecture, inherent power of the chosen logic family and the specific logic design rules used.

The following explanations and examples point out some differences in machines not reflected in clock rates.

1. Difference in Architecture

   a) Size of machine: number of gates. No one expects an 8080 microprocessor to match the power of, say, a middle level 370 machine even if they have comparable clock rates (in fact they do). (The number of gates in a machine is often very difficult to quantify as will be discussed later.)

   b) Concurrency: How much of the machine is in use, on average, during a small unit time. For example, a machine with a 10-ns clock but only 30% of the machine in use at any one time, is very likely slower in overall throughput than a machine with a 20-ns clock but with 70% of its gates in use at a time.

2. Differences in Logic Family

It may seem obvious that the clock rate of a computer should be set by the number of gates that must be traversed to accomplish some basic function. This is not true and one of the reasons is the difficulty in determining what to call a gate. For example most variations of ECL allow both the true and complement results to be generated by most gate elements at no cost in size or speed. This contrasts with other logic families, e.g., TTL, that require another gate function to produce the logical complement of a function. This extra gate appears generally in series to some signal and thus slows down the machine. Even if the gate can be put in a parallel structure to reduce the total delay, the machine may still be slower because the larger number of gates may increase the physical size of the machine.

Another feature of most ECL families is the ability to tie the outputs of gates together. This accomplishes an 'AND' or 'OR' function at virtually no cost in logic delay and no gate cost because the connection is just wire.

The point here is that even with equal clock rates, a computer may have higher performance because it is able to do more in a clock cycle than another computer using similar architectures and numbers of gates because it is able to get more done per unit time no matter what the unit time -- just because of differences in the logical power of differing logic families. Many other attributes, unmentioned, also will affect the logical power available to a designer -- fan-in and fan-out limits, packaging, etc., etc.

3. Design Ground Rules

In, the design of high speed computers, a conflict exists in the interplay for fastest speed versus fastest throughput. Fastest speed means the minimum time for completion of a function, a multiply for example. The fastest throughput means the most results for a function, per unit time. These two requirements are not the same! To satisfy the first design requirement a designer wants the clock cycle to be relatively long. This is to reduce the overhead caused by addition of gates that do not contribute to computation but are required for shorter clock periods. The throughput of a set of logic can be increased by reducing the number of levels of logic allowed between clocked latches. For example, assume that to perform a particular function, such as an addition, 9 levels of logic are required. A designer can choose a clock period of 16 ns, say, and can perform the addition in one clock cycle. Another designer, using the same logic, may choose to limit the number of logic levels between clocked ranks to 7 in order to have a faster clock (say 13 ns). The adder design then requires a clocked latch rank somewhere within the adder. This results in two things:

a) Addition now takes <u>two</u> cycles: 26 ns.

b) The number of gates in the second design is greater, resulting in greater logic cost and design time.

Of course the second adder design has higher throughput, i.e., it can accept a new input every 13 ns instead of every 16 ns as in the first design.

The set of design ground rules chosen depends on many things; among them are: cost, size, power, speed requirements, etc.

As can be seen from the above, the raw comparison of clock rates is a vast oversimplification of the question 'how fast is it?' A whole array of other questions must be asked at the same time.

APPENDIX B

IMPLICIT LEFT-HAND-SIDE SUBROUTINES

```
      SUBROUTINE STEP
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH
     1  ,DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ
     2  ,RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3,LSL,JSL
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(24,30,6,30)
      COMMON/VAR0/S(24,30,5,30)
      COMMON/VAR1/X(24,30,30),Y(24,30,30),Z(24,30,30)
      COMMON /VAR3/P(120,30),XX(4),YY(4),ZZ(4)
      COMMON/COUNT/NC,NC1
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F
      LEVEL 2 Q,S,X,Y,Z
      DYNAMIC A,B,C,D,S,SD,XX(4),YY(4),ZZ(4),XKL ,YKL
      DYNAMIC F,F1(5),F2,S1(5)
      DYNAMIC ZKL
      DYNAMIC XJL ,YJL ,ZJL ,XKJ ,YKJ ,ZKJ
      DYNAMIC QT1,QT2,QT3,QT4,QT5,TV
      DYNAMIC RJ,RR,U,V,W,UU,UT,C1,C2,C3,C4,C5,C6,C7
      DYNAMIC RMJ,RF,XK,YK,ZK,XL,YL,ZL,XJ,YJ,ZJ
      DYNAMIC DPLUS(5,5),DMIN(5,5),QDES(8,2),XYZ(3,2)
      DYNAMIC Q1,Q2,Q3,Q4,Q5
      DYNAMIC X1,Y1,Z1
      INTEGER QINPOS,QOUTPOS,XYZPOS,FDESC(5),SDESC(5)
C
      CALL BC
      CALL RHS
      CALL SMOOTH
C
C  COMPUTE L2 RESIDUAL
C
C   THE CONDITIONAL BRANCH CODE
C***         IF(NC.EQ.1) GO TO 5
C***         IF(NC-(NC/10)*10)5,5,6 .
C***       5 CONTINUE
CAN BE REPLACED BY

      IF(MOD(NC,10).NE.0)GO TO 6

      RESID = 0.0
      KMH = KM
      LMH = LM
      DO 10 N = 1,5
      DO 10 L = 1,LMH
      DO 10 K = 1,KMH
      DO 10 J=2,JM
   10 RESID = RESID+S(K,L,N,J)**2
      RESID = RESID/((JM-1)*(KMH-1)*(LMH-1))
      RESID = SQRT(RESID)/(DT+.00005)
C
C CAN BE VECTORIZED AUTOMATICALLY,HOWEVER THE THROUGHPUT OF THIS LOCAL
C  LOOP WILL BE LIMITED TO THE BANDWIDTH OF THE INTERMEDIATE STORAGE
      WRITE(6,100) NC,RESID
  100 FORMAT(1H0,3HN= ,I5,3X,13HL2 RESIDUAL= ,F16.8)
    6 CONTINUE
C
C
      RM = SMU
      C8 = 1.+2.*RM
      GAM2 = 2.-GAMMA
C   THE NESTED DO LOOP:
C***       DO 20 L = 2,LM
```

```
C***     DO 20 K = 2,KM
C   CAN BE VECTORIZED IN SEGMENTS AS FOLLOWS:
C
C
C***FILTRX
C
      JA=2
      JB=JMAX-1
      DO 1000 L=2,LMAX,LSL
C   LSL IS THE NUMBER OF SLICES(COLUMNS) IN L PER PROCESSING PASS
      LSM=LSL-1
C
      DO 5 N=1,5
      DEFINE (F1(N),F(2:KMAX-1,L:L+LSM,N,2:JMAX-1))
 5    DEFINE (S1(N),S(2:KMAX-1,L:L+LSM,N,2:JMAX-1))
      Q1=Q(*,L:L+LSM,1,2:JMAX-1)
      Q2=Q(*,L:L+LSM,2,2:JMAX-1).
      Q3=Q(*,L:L+LSM,3,2:JMAX-1)
      Q4=Q(*,L:L+LSM,4,2:JMAX-1)
      Q5=Q(*,L:L+LSM,5,2:JMAX-1)
      RJ=Q(2:KMAX-1,L:L+LSM,6,*)
      XKL=X(*,L-1:L+LSM+1,2:JMAX-1)
      YKL=Y(*,L-1:L+LSM+1,2:JMAX-1)
      ZKL=Z(*,L-1:L+LSM+1,2:JMAX-1)
      XK=(XKL(3:KMAX,2:LSL+1,*)-XKL(1:KMAX-2,2:LSL+1,*))*DY2
      YK=(YKL(3:KMAX,2:LSL+1,*)-YKL(1:KMAX-2,2:LSL+1,*))*DY2
      ZK=(ZKL(3:KMAX,2:LSL+1,*)-ZKL(1:KMAX-2,2:LSL+1,*))*DY2
      XL=(XKL(*,3:LSL,*)-XKL(*,1:LSL-2,*))*DZ2
      YL=(YKL(*,3:LSL,*)-YKL(*,1:LSL-2,*))*DZ2
      ZL=(ZKL(*,3:LSL,*)-ZKL(*,1:LSL-2,*))*DZ2
      XX(1)=(YK*ZL-ZK*YL)*RJ(*,*,2:JMAX-1)
      XX(2)=(ZK*XL-XK*ZL)*RJ(*,*,2:JMAX-1)
      XX(3)=(XK*YL-YK*XL)*RJ(*,*,2:JMAX-1)
      XX(4)=-OMEGA*(ZKL(2:KMAX-1,1:LSL,2:JMAX-1)*XX(2)
     1    -YKL(2:KMAX-1,1:LSL,2:JMAX-1)*XX(3))
      D(1,2) =XX(1)*HDX
      D(1,3) =XX(2)*HDX
      D(1,4) =XX(3)*HDX
      D(1,1) =XX(4)*HDX
C
C********AMATRX
C
      RR=1/Q1
      U=RR*Q2
      V=RR*Q3
      W=RR*Q4
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)
      UT = U**2+V**2+W**2
      C1 = GAMI*UT*.5
      C2=RR*GAMMA*Q5
      C3=C2-C1
      C4=D(1,1)+UU
      C5=GAMI*U
      C6=GAMI*V
      C7=GAMI*W
      DEFINE (D(1,5),(1:KMAX-2,1:LSL-2,1:JMAX-2))
      D(1,5)=0
      D(2,1) = D(1,2)*C1-U*UU
      D(2,2) = C4+D(1,2)*GAM2*U
      D(2,3) = -D(1,2)*C6+D(1,3)*U
      D(2,4) = -D(1,2)*C7+D(1,4)*U
      D(2,5) = D(1,2)*GAMI
      D(3,1) = D(1,3)*C1-V*UU
```

```
      D(3,2)  = D(1,2)*V-D(1,3)*C5
      D(3,3)  = C4+D(1,3)*GAM2*V
      D(3,4)  = -D(1,3)*C7+D(1,4)*V
      D(3,5)  = D(1,3)*GAMI
      D(4,1)  = D(1,4)*C1-W*UU
      D(4,2)  = D(1,2)*W-D(1,4)*C5
      D(4,3)  = D(1,3)*W-D(1,4)*C6
      D(4,4)  = C4+D(1,4)*GAM2*W
      D(4,5)  = D(1,4)*GAMI
      D(5,1)  = (-C2+2.*C1)*UU
      D(5,2)  = D(1,2)*C3-C5*UU
      D(5,3)  = D(1,3)*C3-C6*UU
      D(5,4)  = D(1,4)*C3-C7*UU
      D(5,5)  = D(1,1)+GAMMA*UU
C
C******END OF AMATRX
C
      RHJ=RM/RJ(*,*,2:JMAX-1)
      RR=RHJ*RJ(*,*,1:JMAX-2)
      RF=RMJ*RJ(*,*,3:JMAX)
      DO 23 N=1,5
      DO 22 M=1,5
      DEFINE (B(N,M),(1:KMAX-2,1:LSM,1:JMAX-2)
      DEFINE (D1,D(N,M))
      A(N,M)=-D1(*,*,1:JMAX-2)
      C(N,M)=D1(*,*,3:JMAX)
 22   B(N,M)=0
      A(N,N) = A(N,N)-RR
      B(N,N) = C8
      C(N,N) = C(N,N)-RF
 23    F1(N)=S1(N)
C
C*****END OF FILTRX
C
C   S MUST BE ZERO ON B.C.
C
C
      CALL BTRI(2,JM)
      DO 24 N=1,5
      S1(N)=F1(N)
 24   CONTINUE
C
C
 1000 CONTINUE
C
C******FILTRY
C
      KA = 2
      KB = KMAX-1
      JSM = JSL-1
      DO 2000 J=2,JMAX,JSL
      DO 6 N=1,5
      DEFINE (F1(N),F(2:LMAX-1,J:J+JSM,N,2:KMAX-1))
  6   DEFINE (S1(N),F(2:LMAX-1,J:J+JSM,N,2:KMAX-1))
C
      RJ = RJT(2:LMAX-1,J:J+JSM,2:KMAX)
      XJL = XJT
      YJL = YJLT
      ZJL = ZJLT
      Q1 = Q1T
      Q2 = Q2T
      Q3 = Q3T
      Q4 = Q4T
```

```
      Q5 = Q5T
C
C
      DO 9 N=1,5
      DO 9 K=1,KMAX
      DEFINE (F2,F1(N))
 9    F2(*,*,K)=S(K,2:LMAX-1,N,J:J+JSM)
C  THIS ACCOMPLISHES THE MOVE OF S TO THE F ARRAY
C
      XJ=(XJL(2:LMAX-1,3:JSL+2,2:KMAX-1)-XJL(2:LMAX-1,1:JSL,2:KMAX-1))
     1  *DX2
      YJ=(YJL(2:LMAX-1,3:JSL+2,2:KMAX-1)-XJL(2:LMAX-1,1:JSL,2:KMAX-1))
     1   *DX2
      ZJ=(ZJL(2:LMAX-1,3:JSL+2,2:KMAX-1)-XJL(2:LMAX-1,1:JSL,2:KMAX-1))
     1   *DY2
      XL=(XJL(3:LMAX,2:JSL+1,2:KB)-XJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2
      YL=(YJL(3:LMAX,2:JSL+1,2:KB)-YJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2

      ZL=(ZJL(3:LMAX,2:JSL+1,2:KB)-ZJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2
      YY(1)=(ZJ*YL-YJ*ZL)*RJ(*,*,2:KMAX-1)
      YY(2)=(XJ*ZL-XL*ZJ)*RJ(*,*,2:KMAX-1)
      YY(3)=(YJ*XL-XJ*YL)*RJ(*,*,2:KMAX-1)
      YY(4)=-OMEGA*(ZJL(2:LMAX-1,2:JSM,2:KMAX-1)*YY(2)
     1               YJL(2:LMAX-1,2:JSM,2:KMAX-1)*YY(3))
      D(1,2) =YY(1)*HDY
      D(1,3) =YY(2)*HDY
      D(1,4) =YY(3)*HDY
      D(1,1) =YY(4)*HDY
C
C******AMATRX
      RR=1/Q1
      U=RR*Q2
      V=RR*Q3
      W=RR*Q4
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)
      UT = U**2+V**2+W**2
      C1 = GAMI*UT*.5
      C2=RR*GAMMA*Q5
      C3=C2-C1
      C4=D(1,1)+UU
      C5=GAMI*U
      C6=GAMI*V
      C7=GAMI*W
      DEFINE (D(1,5),(1:LMAX-2,1:JSL,1:KMAX-2))
      D(1,5) = 0.
      D(2,1) = D(1,2)*C1-U*UU
      D(2,2) = C4+D(1,2)*GAM2*U
      D(2,3) = -D(1,2)*C6+D(1,3)*U
      D(2,4) = -D(1,2)*C7+D(1,4)*U
      D(2,5) = D(1,2)*GAMI
      D(3,1) = D(1,3)*C1-V*UU
      D(3,2) = D(1,2)*V-D(1,3)*C5
      D(3,3) = C4+D(1,3)*GAM2*V
      D(3,4) = -D(1,3)*C7+D(1,4)*V
      D(3,5) = D(1,3)*GAMI
      D(4,1) = D(1,4)*C1-W*UU
      D(4,2) = D(1,2)*W-D(1,4)*C5
      D(4,3) = D(1,3)*W-D(1,4)*C6
      D(4,4) = C4+D(1,4)*GAM2*W
      D(4,5) = D(1,4)*GAMI
      D(5,1) = (-C2+2.*C1)*UU
      D(5,2) = D(1,2)*C3-C5*UU
      D(5,3) = D(1,3)*C3-C6*UU
```

```
      D(5,4) = D(1,4)*C3-C7*UU
      D(5,5) = D(1,1)+GAMMA*UU
C
C*****END OF AMATRX
C
C    RJ IS RETAINED FROM THE DIFFERENCING CALCULATION
      RMJ=RM*RJ(*,*,2:KMAX-1)
      RR=RMJ*RJ(*,*,1:KMAX-2)
      RF=RMJ*RJ(*,*,3:KMAX)
      DO 34 N=1,5
      DO 33 M=1,5
      DEFINE (D1,D(N,M))
      DEFINE(B(N,M),(1:LMAX-2,1:JMAX-2,1:KMAX-2))
C
C  NOTE THAT B HAS CHANGED SHAPE AS WE ARE DEALING WITH
C     THE TRANSPOSED MESH FOR THIS SWEEP
      A(N,M)=-D1(*,*,1:KMAX-2)
      B(N,M)=0
 33   C(N,M)=D1(*,*,3:KMAX)
      A(N,N) = A(N,N)-RR
      B(N,N) = C8
      C(N,N) = C(N,N)-RF
 34   CONTINUE
C   F HAS BEEN PREMOVED AT TOP OF LOOP
C
C*******END OF FILTRY
C
      CALL BTRI(2,KM)
      DO 31 N=1,5
      DEFINE (S2,S1(N)),(F2,F1(N))
      DO 31 K=2,K8
      S(K,2:LMAX-1,N,J:J+JSM) = F2(2:LMAX-1,1:JSL-2,K)
 31   CONTINUE
C
C
C
 2000 CONTINUE
C******FILTRZ
      DEFINE (RJ,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q1,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q2,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q3,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q4,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q5,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (XKJ,(1:KMAX,1:JSL+2,1:LMAX))
      DEFINE (YKL,(1:KMAX,1:JSL+2,1:LMAX))
      DEFINE (YKL,(1:KMAX,1:JSL+2,1:LMAX))
      LA =2
      LB =LMAX-1
      JSM = JSL-1
      DO 3000 J=2,JMAX,JSL
C
      DO 61 N=1,5
   61 DEFINE (F1(N),(F(2:KMAX-1,J:J+JSM,N,2:LMAX-1)
      DO 81 L=1,LMAX
      RJ(1:KMAX-2,1:JSL,L)=Q(2:KMAX-1,L,6,J:J+JSM)
      XKJ(1:KMAX,1:JSL+2,L)=X(1:KMAX,L,J-1:J+JSL)
      YKJ(1:KMAX,1:JSL+2,L)=Y(1:KMAX,L,J-1:J+JSL)
      ZKJ(1:KMAX,1:JSL+2,L)=Z(1:KMAX,L,J-1:J+JSL)
      Q1(1:KMAX,1:JSL,L)=Q(2:KMAX-1,L,1,J:J+JSM)
      Q2(1:KMAX,1:JSL,L)=Q(2:KMAX-1,L,2,J:J+JSM)
      Q3(1:KMAX,1:JSL,L)=Q(2:KMAX-1,L,3,J:J+JSM)
      Q4(1:KMAX,1:JSL,L)=Q(2:KMAX-1,L,4,J:J+JSM)
```

```
      Q5(1:KMAX,1:JSL,L)=Q(2:KMAX-1,L,5,J:J+JSM)
C
C
 81   CONTINUE
      DO 91 N=1,5
      DO 91 L=1,LMAX
      DEFINE (F2;F1(N))
 91   F2(*,*,L)=S(2:KMAX-1,L,N,J:J+JSM)
      XK=(XKJ(3:KMAX,2:JSL+1,2:LB)-XKJ(1:KMAX-2,2:JSL+1,2:LB))*DY2
      YK=(YKJ(3:KMAX,2:JSL+1,2:LB)-YKJ(1:KMAX-2,2:JSL+1,2:LB))*DY2
      ZK=(ZKJ(3:KMAX,2:JSL+1,2:LB)-ZKJ(1:KMAX-2,2:JSL+1,2:LB))*DY2
      XJ=(XKJ(2:KMAX-1,3:JSL+2,2:LB)-XKJ(2:KMAX-1,1:JSL,2:LB))*DX2
      YJ=(YKJ(2:KMAX-1,3:JSL+2,2:LB)-YKJ(2:KMAX-1,1:JSL,2:LB))*DX2
      ZJ=(ZKJ(2:KMAX-1,3:JSL+2,2:LB)-ZKJ(2:KMAX-1,1:JSL,2:LB))*DX2
      ZZ(1)=(YJ*ZK-ZJ*YK)*RJ
      ZZ(2)=(XK*ZJ-XJ*ZK)*RJ
      ZZ(3)=(XJ*YK-YJ*XK)*RJ
      ZZ(4)=-OMEGA*(ZKJ(2:KMAX-1,2:JSM,2:LB)*ZZ(2)
     1             -YKJ(2:KMAX-1,2:JSM,2:LB)*ZZ(3))

      D(1,2) =ZZ(1)*HDZ
      D(1,3) =ZZ(2)*HDZ
      D(1,4) =ZZ(3)*HDZ
      D(1,1) =ZZ(4)*HDZ
C
C******AMATRX
      RR=1./Q1
      U=RR*Q2
      V=RR*Q3
      W=RR*Q4
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)
      UT = U**2+V**2+W**2
      C1 = GAMI*UT*.5
      C2=RR*Q5*GAMMA
      C3=C2-C1
      C4=D(1,1)+UU
      C5=GAMI*U
      C6=GAMI*V
      C7=GAMI*W
      DEFINE(D(1,5),(1:KMAX-2,1:JSL,1:LMAX-2))
      D(1,5) = 0.
      D(2,1) = D(1,2)*C1-U*UU
      D(2,2) = C4+D(1,2)*GAM2*U
      D(2,3) = -D(1,2)*C6+D(1,3)*U
      D(2,4) = -D(1,2)*C7+D(1,4)*U
      D(2,5) = D(1,2)*GAMI
      D(3,1) = D(1,3)*C1-V*UU
      D(3,2) = D(1,2)*V-D(1,3)*C5
      D(3,3) = C4+D(1,3)*GAM2*V
      D(3,4) = -D(1,3)*C7+D(1,4)*V
      D(3,5) = D(1,3)*GAMI
      D(4,1) = D(1,4)*C1-W*UU
      D(4,2) = D(1,2)*W-D(1,4)*C5
      D(4,3) = D(1,3)*W-D(1,4)*C6
      D(4,4) = C4+D(1,4)*GAM2*W
      D(4,5) = D(1,4)*GAMI
      D(5,1) = (-C2+2.*C1)*UU
      D(5,3) = D(1,3)*C3-C6*UU
      D(5,4) = D(1,4)*C3-C7*UU
      D(5,5) = D(1,1)+GAMMA*UU
C
C*****END OF AMATRX
C
```

```
C     RJ IS RETAINED FROM DIFFERENCING OPERATION
      RMJ = RM/RJ(*,*,2:LMAX-1)
      RF = RMJ*RJ(*,*,3:LMAX)
      RR = RMJ*RJ(*,*,1:LMAX-2)
      DO 43 N=1,5
      DO 42 M=1,5
      DEFINE(D1,D(N,M))
      DEFINE(B(N,M),(1:KMAX-2,1:JSL,1:LMAX-2))
      A(N,M)=-D1(*,*,1:LMAX-2)
      B(N,M)=0
 42   C(N,M)=D1(*,*,3:LMAX)
      A(N,N) = A(N,N)-RR
      B(N,N) = CB
      C(N,N) = C(N,N)-RF
 43   CONTINUE
C
C*******END OF FILTRZ
C
      IF(INVISC.EQ.1) CALL VISMAT(J,K)
      CALL BTRI(2,LM)
      Q1=Q1+F(1)
      Q2=Q2+F(2)
      Q3=Q3+F(3)
      Q4=Q4+F(4)
      Q5=Q5+F(5)
      DO 44 L=2,LMAX-1
      Q(2:KMAX-1,L,1,J:J+JSM)=Q1(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,2,J:J+JSM)=Q2(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,3,J:J+JSM)=Q3(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,4,J:J+JSM)=Q4(1:KMAX-2,1:JSL-2,L)
 44   Q(2:KMAX-1,L,5,J:J+JSM)=Q5(1:KMAX-2,1:JSL-2,L)
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE BTRI(ILA,IUA)
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F(5)
      DYNAMIC H(5,5),A,B,C,D,F
      DYNAMIC L11,L21,L22,L31,L32,L33,L41,L42,L43,L44
      DYNAMIC L51,L52,L53,L54,L55
      DYNAMIC U11,U21,U22,U31,U32,U33,U41,U42,U43,U44
      DYNAMIC U51,U52,U53,U54,U55
      DYNAMIC D1,D2,D3,D4,D5
      DYNAMIC U12,U13,U14,U15,U23,U24,U25,U34,U35,U45
      DYNAMIC A1(5,5),B1(5,5),C1(5,5),F1(5)
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,L51,L52,L53,L54,L55
      IL=ILA
      IU=IUA
      IS=IL+1
      IE=IU-1
      DO 1 N=1,5
      DEFINE (F1(N),F(N)(*,*,1))
      DO 1 M=1,5
      DEFINE (B1(N,M),B(N,M)(*,*,1))
      DEFINE (C1(N,M),C(N,M)(*,*,1))
      INSERT LUDEC
      L11=1./B (1,1)
      L21=B1(2,1)
      U12=B1(1,2)*L11
      L22=1./(B1(2,2)-L21*U12)
      U13=B1(1,3)*L11
      U14 = B1(1,4)*L11
      U15=B1(1,5)*L11
      L31=B1(3,1)
```

```
      L32=B1(3,2)-L31*U12
      U23=(B1(2,3)-L21*U13)*L22
      L33=1./(B1(3,3)-U13*L31-U23*L32)
      U24=(B1(2,4)-L21*U14)*L22
      U25=(B1(2,5)-L21*U15)*L22
      L41=B1(4,1)
      L42=B1(4,2)-L41*U12
      L43=B1(4,3)-L41*U13-L42*U23
      U34=(B1(3,4)-L31*U14-L32*U24)*L33
      L44=1./(B1(4,4)-U14*L41-U24*L42-U34*L43)
      U35=(B1(3,5)-L31*U15-L32*U25)*L33
      L51=B1(5,1)
      L52=B1(5,2)-L51*U12
      L53=B1(5,3)-L51*U13-L52*U23
      L54=B1(5,4)-L51*U14-L52*U24-L53*U34
      U45=(B1(4,5)-L41*U15-L42*U25-L43*U35)*L44
      L55=1./(B1(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C     COMPUTE LITTLE R S
      D1=L11*F1(1)
      D2=L22*(F1(2)-L21*D1)
      D3=L33*(F1(3)-L31*D1-L32*D2)
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C     COMPUTE BIG R S
      F1(5)=D5
      F1(4)=D4-U45*D5
      F1(3)=D3-U34*F1(4)-U35*D5
      F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5
      F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5
C     COMPUTE C PRIME FOH FIRST ROW
      DO 12 M=1,5
      D1=L11*C1(1,M)
      D2=L22*(C1(2,M)-L21*D1)
      D3=L33*(C1(3,M)-L31*D1-L32*D2)
      D4=L44*(C1(4,M)-L41*D1-L42*D2-L43*D3)
      D5=L55*(C1(5,M)-L51*D1-L52*D2-L53*D3-L54*D4)
      B1(5,M)=D5
      B1(4,M)=D4-U45*D5
      B1(3,M) = D3-U34*B1(4,M)-U35*D5
      B1(2,M) = D2-U23*B1(3,M)-U24*B1(4,M)-U25*D5
12    B1(1,M) = D1-U12*B1(2,M)-U13*B1(3,M)-U14*B1(4,M)-U15*D5
C      COMPUTE B PRIME*BIGR
      DO 13 I=IS,IE
      DO 2 N=1,5
      DEFINE (F1(N),F(N)(*,*,I)),(F2(N),F(N)(*,*,I-1))
      DO 2 M=1,5
      DEFINE (A1(N,M),A(N,M)(*,*,I))
      DEFINE (B1(N,M),B(N,M)(*,*,I)),(B2(N,M),B(N,M)(*,*,I-1))
 2    DEFINE (C1(N,M),C(N,M)(*,*,I))
      DO 14 N=1,5
 14   F1(N)=F1(N)-A1(N,1)*F2(1)-A1(N,2)*F2(2)-A1(N,3)*F2(3)
     *       -A1(N,4)*F2(4)-A1(N,5)*F2(5)
C      COMPUTE B PRIME
      DO 11 N=1,5
      DO 11 M=1,5
 11   H(N,M)=B1(N,M)-A1(N,1)*B2(1,M)-A1(N,2)*B2(2,M)-A1(N,3)*
     *        B2(3,M)-A1(N,4)*B2(4,M)-A1(N,5)*B2(5,M)
C     INSERT LUDEC AGAIN
      L11=1./H(1,1)
      L21=H(2,1)
      U12=H(1,2)*L11
      L22=1./(H(2,2)-L21*U12)
      U13=H(1,3)*L11
```

```
      U14=H(1,4)*L11
      U15=H(1,5)*L11
      L31=H(3,1)
      L32=H(3,2)-L31*U12
      U23=(H(2,3)-L21*U13)*L22
      L33=1./(H(3,3)-U13*L31-U23*L32)
      U24=(H(2,4)-L21*U14)*L22
      U25=(H(2,5)-L21*U15)*L22
      L41=H(4,1)
      L42=H(4,2)-L41*U12
      L43=H(4,3)-L41*U13-L42*U23
      U34=(H(3,4)-L31*U14-L32*U24)*L33
      L44=1./(H(4,4)-U14*L41-U24*L42-U34*L43)
      U35=(H(3,5)-L31*U15-L32*U25)*L33
      L51=H(5,1)
      L52=H(5,2)-L51*U12
      L53=H(5,3)-L51*U13-L52*U23
      L54=H(5,4)-L51*U14-L52*U24-L53*U34
      U45=(H(4,5)-L41*U15-L42*U25-L43*U35)*L44
      L55=1./(H(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C     COMPUTE LITTLE RIS
      D1=L11*F1(1)
      D2=L22*(F1(2)-L21*D1)
      D3=L33*(F1(3)-L31*D1-L32*D2)
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C     COMPUTE BIG RIS
      F1(5)=D5
      F1(4)=D4-U45*D5
      F1(3)=D3-U34*F1(4)-U35*D5
      F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5
      F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5
C     COMPUTE C PRIMES
      DO 15 M=1,5
      D1=L11*C1(1,M)
      D2=L22*(C1(2,M)-L21*D1)
      D3=L33*(C1(3,M)-L31*D1-L32*D2)
      D4=L44*(C1(4,M)-L41*D1-L42*D2-L43*D3)
      D5=L55*(C1(5,M)-L51*D1-L52*D2-L53*D3-L54*D4)
      B1(5,M)=D5
      B1(4,M)=D4-U45*D5
      B1(3,M) = D3-U34*B1(4,M)-U35*D5
      B1(2,M) = D2-U23*B1(3,M)-U24*B1(4,M)-U25*D5
15    B1(1,M) = D1-U12*B1(2,M)-U13*B1(3,M)-U14*B1(4,M)-U15*D5
13    CONTINUE
      I=IU
      DO 3 N=1,5
      DEFINE (F1(N),F(N)(*,*,IU)),(F2(N),F(N)(*,*,IU-1))
      DO 3 M=1,5
      DEFINE (A1(N,M),A(N,M)(*,*,IU))
 3    DEFINE (B1(N,M),B(N,M)(*,*,IU)), (B2(N,M),B(N,M)(*,*,IU-1))
C      COMPUTE B PRIME*BIG R FOR LAST ROW
      DO 17 N=1,5
 17   F1(N)=F1(N)-A1(N,1)*F2(1)-A1(N,2)*F2(2)-A1(N,3)*
     1      F2(3)-A1(N,4)*F2(4)-A1(N,5)*F2(5)
C      COMPUTE B PRIME
      DO 18 N=1,5
      DO 18 M=1,5
 18   H(N,M)=B1(N,M)-A1(N,1)*B2(1,M)-A1(N,2)*B2(2,M)-A1(N,3)*
     *B2(3,M)-A1(N,4)*B2(4,M)-A1(N,5)*B2(5,M)
C     INSERT LUDEC AGAIN
      L11=1./H(1,1)
      L21=H(2,1)
```

```
      U12=H(1,2)*L11
      L22=1./(H(2,2)-L21*U12)
      U13=H(1,3)*L11
      U14=H(1,4)*L11
      U15=H(1,5)*L11
      L31=H(3,1)
      L32=H(3,2)-L31*U12
      U23=(H(2,3)-L21*U13)*L22
      L33=1./(H(3,3)-U13*L31-U23*L32)
      U24=(H(2,4)-L21*U14)*L22
      U25=(H(2,5)-L21*U15)*L22
      L41=H(4,1)
      L42=H(4,2)-L41*U12
      L43=H(4,3)-L41*U13-L42*U23
      U34=(H(3,4)-L31*U14-L32*U24)*L33
      L44=1./(H(4,4)-U14*L41-U24*L42-U34*L43)
      U35=(H(3,5)-L31*U15-L32*U25)*L33
      L51=H(5,1)
      L52=H(5,2)-L51*U12
      L53=H(5,3)-L51*U13-L52*U23
      L54=H(5,4)-L51*U14-L52*U24-L53*U34
      U45=(H(4,5)-L41*U15-L42*U25-L43*U35)*L44
      L55=1./(H(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C     COMPUTE LITTLE RIS
      D1=L11*F1(1)
      D2=L22*(F1(2)-L21*D1)
      D3=L33*(F1(3)-L31*D1-L32*D2)
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C     COMPUTE BIG RIS
      F1(5)=D5
      F1(4)=D4-U45*D5
      F1(3)=D3-U34*F1(4)-U35*D5
      F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5
      F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5
      I=IU
20    I=I-1
      DO 6 N=1,5
      DEFINE (F1(N),F(N)(*,*,I)),(F2(N),F(N)(*,*,I+1))
  6   CONTINUE
      DO 19 N=1,5
 19   F1(N)=F1(N)-F2(1)*B1(N,1)-F2(2)*B1(N,2)-F2(3)*B1(N,3)
     *        -F2(4)*B1(N,4)-F2(5)*B1(N,5)
      IF (I.GT.IL)GOTO20
      RETURN
      END
```

APPENDIX C

IMPLICIT RIGHT-HAND-SIDE SUBROUTINES

```
      SUBROUTINE RHS
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH,
     1  DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ,
     2  RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(243,30,6,30)
      COMMON/VAR0/S(243,30,5,30)
      COMMON/VAR1/X(243,30,30),Y(243,30,30),Z(243,30,30)
      COMMON/VAR3/P(120,30),XX(60,4),YY(60,4),ZZ(60,4)
      DYNAMIC A,B,C,D,S,SD,XX(4),YY(4),ZZ(4),XKL,YKL,ZKL
      DYNAMIC F,F1,F2
      DYNAMIC XJL,YJL,ZJL,XKJ,YKJ,ZKJ
      DYNAMIC RMJ,RF,XK,YK,ZK,XL,YL,ZL,XJ,YJ,ZJ
      LEVEL 2,Q,S,X,Y,Z
      DYNAMIC Q1,Q2,Q3,Q4,Q5,Q6,Q7,Q8
      DYNAMIC X1,Y1,Z1
      COMMON/FLSH/DX2,DY2,DZ2
      COMMON/IDX/LMM,KMM,JMM
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F(5)
      R0 = -HDZ
      DO 1000 J=2,JMAX-1,JSL
      RJ=Q(*,*,6,J:J+JSL)
      XJL=X(*,*,J-1:J+JSL+1)
      YJL=Y(*,*,J-1:J+JSL+1)
      ZJL=Z(*,*,J-1:J+JSL+1)
      Q1=Q(*,*,1,J:J+JSL)
      Q2=Q(*,*,2,J:J+JSL)
      Q3=Q(*,*,3,J:J+JSL)
      Q4=Q(*,*,4,J:J+JSL)
      Q5=Q(*,*,5,J:J+JSL)
      XK=(XJL(3:KMAX+2,*,2:JSL+1)-XJL(-1:KMAX-1,*,2:JSL+1))*DY2
      YK=(YJL(3:KMAX+2,*,2:JSL+1)-YJL(-1:KMAX-1,*,2:JSL+1))*DY2
      ZK=(ZJL(3:KMAX+2,*,2:JSL+1)-ZJL(-1:KMAX-1,*,2:JSL+1))*DY2
      XJ=(XJL(*,*,3:JSL+2)-XJL(*,*,1:JSL))*DX2
      YJ=(YJL(*,*,3:JSL+2)-YJL(*,*,1:JSL))*DX2
      ZJ=(ZJL(*,*,3:JSL+2)-ZJL(*,*,1:JSL))*DX2
      ZZ(1)=(YJ*ZK-ZJ*YK)*RJ
      ZZ(2)=(XK*ZJ-XJ*ZK)*RJ
      ZZ(3)=(XJ*YK-YJ*XK)*RJ
      ZZ(4)=-OMEGA*(ZKJ(*,*,2:JSL+1)*ZZ(2)-YKJ(*,*,2:JSL+1)*ZZ(3))
C
C*******FLUXVE
C
      RR = 1./Q1
      U = Q2*RR
      V = Q3*RR
      W = Q4*RR
      QS = ZZ(4)+ZZ(1)*U+ZZ(2)*V+ZZ(3)*W
      PP = GAMI*(Q5-.5*Q1*(U*U+V*V+W*W))
      F1(1) = Q1*QS
      F1(2) = Q2*QS+ZZ(1)*PP
      F1(3) = Q3*QS+ZZ(2)*PP
      F1(4) = Q4*QS+ZZ(3)*PP
      F1(5) = (Q5+PP)*QS-ZZ(4)*PP
C
C*******END OF FLUXVE
C
      DO 20 N=1,5
      DEFINE (F2,F1(N)), (S2,S1(N))
      S2(*,2:LMAX+1,*)=(F2(*,3:LMAX+2,*)-F2(*,1:LMAX,*))*R0
 20   CONTINUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

C-3

```
C
C
      R0 = -HDY
      XJ=(XJL(*,*,3:JSL+2)-XJL(*,*,1:JSL))*DX2
      YJ=(YJL(*,*,3:JSL+2)-YJL(*,*,1:JSL))*DX2
      ZJ=(ZJL(*,*,3:JSL+2)-ZJL(*,*,1:JSL))*DX2
      XL=(XJL(*,2:LMAX+1,2:JSL+1)-XJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      YL=(YJL(*,2:LMAX+1,2:JSL+1)-YJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      ZL=(ZJL(*,2:LMAX+1,2:JSL+1)-ZJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      YY(1)=(ZJ*YL-YJ*ZL)*RJ
      YY(2)=(XJ*ZL-XL*ZJ)*RJ
      YY(3)=(YJ*XL-XJ*YL)*RJ
      YY(4)=-OMEGA*(ZJL(*,*,2:JSL+1)*YY(2)-YJL(*,*,2:JSL+1)*YY(3))
C
C*******FLUXVE
C
      QS = YY(4)+YY(1)*U+YY(2)*V+YY(3)*W
      PP = GAMI*(Q5-.5*Q1*(U*U+V*V+W*W))
      F1(1)  = Q1*QS
      F1(2)  = Q2*QS+YY(1)*PP
      F1(3)  = Q3*QS+YY(2)*PP
      F1(4)  = Q4*QS+YY(3)*PP
      F1(5)  = (Q5+PP)*QS-YY(4)*PP
C
C*******END OF FLUXVE
C
      DO 21 N=1,5
      DEFINE (F2,F1(N)), (S12,S1(N))
      S3(2:KMAX+1,*,*)=(F2(3:KMAX+2,*,*)-F2(1:KMAX,*,*))*R0
      S2=S2+S3
 21   CONTINUE
C
C
      R0 = -HDX
      XK=(XJL(2:KMAX+1,*,2:JSL+1)-XJL(-1:KMAX-1,*,2:JSL+1))*DY2
      YK=(YJL(2:KMAX+1,*,2:JSL+1)-YJL(-1:KMAX-1,*,2:JSL+1))*DY2
      ZK=(ZJL(2:KMAX+1,*,2:JSL+1)-ZJL(-1:KMAX-1,*,2:JSL+1))*DY2
      XL=(XJL(*,2:LMAX+1,2:JSL+1)-XJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      YL=(YJL(*,2:LMAX+1,2:JSL+1)-YJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      ZL=(ZJL(*,2:LMAX+1,2:JSL+1)-ZJL(*,-1:LMAX-1,2:JSL+1))*DZ2
      XX(1)=(YK*ZL-ZK*YL)*RJ
      XX(2)=(ZK*XL-XK*ZL)*RJ
      XX(3)=(XK*YL-YK*XL)*RJ
      XX(4)=-OMEGA*(ZKL(*,*,2:JSL)*XX(2)-YKL(*,*,2:JSL)*XX(3))
C
C*******FLUXVE
C
      QS = XX(4)+XX(1)*U+XX(2)*V+XX(3)*W
      PP = GAMI*(Q5-.5*Q1*(U*U+V*V+W*W))
      F1(1)  = Q1*QS
      F1(2)  = Q2*QS+XX(1)*PP
      F1(3)  = Q3*QS+XX(2)*PP
      F1(4)  = Q4*QS+XX(3)*PP
      F1(5)  = (Q5+PP)*QS-yX(4)*PP
C
C*******END OF FLUXVE
C
      DO 22 N=1,5
      DEFINE (F2,F1(N)), (S2,S1(N))
      S3(*,*,2:JSL+1) = (F2(*,*,3:JSL+2)-F2(*,*,1:JSL)) * R0
      S2=S2+S3
 22   CONTINUE
      IF(INVISC.EQ.1) CALL VISRHS
```

```
      DO 1000 N=1,5
      DEFINE (S4,S(N))
      S4(2:KMAX-1,2:LMAX-1,J:J+JSL)=S2(N)
1000  CONTINUE
      RETURN
      END
      SUBROUTINE VISMAT(JA,KA)
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH,
     1  DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ,
     2  RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(243,30,6,30)
      COMMON/VAR0/S(243,30,5,30)
      COMMON/VAR1/X(243,30,30),Y(243,30,30),Z(243,30,30)
      COMMON/VAR3/P(120,30),XX(4),YY(4),ZZ(4)
      LEVEL 2,Q,S,X,Y,Z
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F(5)
      COMMON/COUNT/NC,NC1
      COMMON TURMU(243,30,30)
      DIMENSION DU(5,5)
      DYNAMIC D,DU,S0,S1,S2,S3,S4,S5,S6,C0,C1,C2,C3,C4,C5,C6
      DYNAMIC A,B,C,RR,RJ,D1,DU1
      DYNAMIC XX,YY,ZZ, RJ,Q5,RR, U,V,W
      COMMON/MOR/ L, RJ,Q5,RR, U,V,W
      J=JA
      K=KA
      GKPR = GAMMA/PR
      PRTR = PR/0.9
      DRE = -HD/(RE*DZ1**2)
      R3 = 1./3.
      TURM = TURMU(K,L,J)
      VNU = RMUE*TURM
      GKAP = RMUE*PRTR*TURM
      S0 = (ZZ(1)**2+ZZ(2)**2+ZZ(3)**2)*RJ
      S1 = (S0 +R3*ZZ(1)**2*RJ)*VNU
      S2 = (S0 +R3*ZZ(2)**2*RJ)*VNU
      S3 = (S0 +R3*ZZ(3)**2*RJ)*VNU
      S4 = R3*ZZ(1)*ZZ(2)*RJ*VNU
      S5 = R3*ZZ(1)*ZZ(3)*RJ*VNU
      S6 = R3*ZZ(2)*ZZ(3)*RJ*VNU
      S0 = S0 *GKPR*GKAP
      E  = Q5*RR
      C0 = (S0(*,*,2:LMAX)+S0(*,*,1:LMAX-1))
      C1 = (S1(*,*,2:LMAX)+S1(*,*,1:LMAX-1))
      C2 = (S2(*,*,2:LMAX)+S2(*,*,1:LMAX-1))
      C3 = (S3(*,*,2:LMAX)+S3(*,*,1:LMAX-1))
      C4 = (S4(*,*,2:LMAX)+S4(*,*,1:LMAX-1))
      C5 = (S5(*,*,2:LMAX)+S5(*,*,1:LMAX-1))
      C6 = (S6(*,*,2:LMAX)+S6(*,*,1:LMAX-1))
      DEFINE(D(2,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      DEFINE(DU(2,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      DEFINE(D(3,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      DEFINE(DU(3,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      DEFINE(D(4,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      DEFINE(DU(4,5),(1:KMAX-2,1:JSL,1:LMAX-1))
      D(2,1) = -(C1*U(*,*,1:LMAX-1)+C4*V(*,*,1:LMAX-1)+C5*
     1   W(*,*,1:LMAX-1))*RR(*,*,1:LMAX-1)
      DU(2,1)= -(C1*U(*,*,2:LMAX)+C4*V(*,*,2:LMAX)+C5*
     1   W(*,*,2:LMAX))*RR(*,*,1:LMAX-1)
      D(2,2) = C1*RR(*,*,1:LMAX-1)
      DU(2,2) = C1*RR(*,*,2:LMAX)
```

```
      D(2,3) = C4*RR(*,*,1;LMAX-1)
      DU(2,3) = C4*RR(*,*,2;LMAX)
      D(2,4) = C5*RR(*,*,1;LMAX-1)
      DU(2,4) = C5*RR(*,*,2;LMAX)
      D(2,5) = 0.0
      DU(2,5) = 0.0
      D(3,1) = -(C2*V(*,*,1;LMAX-1)+C4*U(*,*,1;LMAX-1)+C6*
     1   W(*,*,1;LMAX-1))*RR(*,*,1;LMAX-1)
      DU(3,1)= -(C2*V(*,*,2;LMAX)+C4*U(*,*,2;LMAX)+C6*
     1   W(*,*,2;LMAX))*RR(*,*,1;LMAX-1)
      D(3,2) = C4*RR(*,*,1;LMAX-1)
      DU(3,2) = C4*RR(*,*,2;LMAX)
      D(3,3) = C2*RR(*,*,1;LMAX-1)
      DU(3,3) = C2*RR(*,*,2;LMAX)
      D(3,4) = C6*RR(*,*,1;LMAX-1)
      DU(3,4) = C6*RR(*,*,2;LMAX)
      D(3,5) = 0.0
      DU(3,5) = 0.0
      D(4,1) = -(C3*W(*,*,1;LMAX-1)+C5*U(*,*,1;LMAX-1)+C6*
     1   V(*,*,1;LMAX-1))*RR(*,*,1;LMAX-1)
      DU(4,1) =-(C3*W(*,*,2;LMAX)+C5*U(*,*,2;LMAX)+C6*
     1   V(*,*,2;LMAX))*RR(*,*,1;LMAX-1)
      D(4,2) = C5*RR(*,*,1;LMAX-1)
      DU(4,2) = C5*RR(*,*,2;LMAX)
      D(4,3) = C6*RR(*,*,1;LMAX-1)
      DU(4,3) = C6*RR(*,*,2;LMAX)
      D(4,4) = C3*RR(*,*,1;LMAX-1)
      DU(4,4) = C3*RR(*,*,2;LMAX)
      D(4,5) = 0.0
      DU(4,5) = 0.0
      D(5,1) = -((C1-C0)*U(*,*,1;LMAX-1)**2+(C2-C0)*V(*,*,1;LMAX-1)**2
     1   +(C3-C0)*W(*,*,1;LMAX-1)**2+2.*C4*U(*,*,1;LMAX-1)*
     2   V(*,*,1;LMAX-1)+2.*C5*U(*,*,1;LMAX-1)*W(*,*,1;LMAX-1)+
     3   2.*C6*V(*,*,1;LMAX-1)*W(*,*,1;LMAX-1)+
     4   C0*E(*,*,1;LMAX-1))*RR(*,*,1;LMAX-1)
      DU(5,1) = -((C1-C0)*U **2+(C2-C0)*V **2+(C3-C0)*W **2+
     1   2.*C4*U *V +2.*C5*U *W +2.*C6*V *W +C0*E )*RR(*,*,2;LMAX)
      D(5,2) = ((C1-C0)*U(*,*,1;LMAX-1)+C4*V(*,*,1;LMAX-1)+C5*
     1   W(*,*,1;LMAX-1))*RR(1;LMAX-1)
      DU(5,2)= ((C1-C0)*U(*,*,2;LMAX)+C4*V(*,*,2;LMAX)+C5*
     1   W(*,*,2;LMAX))*RR(*,*,1;LMAX-1)
      D(5,3) = ((C2-C0)*V(*,*,1;LMAX-1)+C4*U(*,*,1;LMAX-1)+C6*
     1   W(*,*,1;LMAX-1))*RR(*,*,1;LMAX-1)
      DU(5,3)= ((C2-C0)*V(*,*,2;LMAX)+C4*U(*,*,2;LMAX)+C6*
     1   W(*,*,2;LMAX))*RR(*,*,1;LMAX-1)
      D(5,4) = ((C3-C0)*W(*,*,1;LMAX-1)+C5*U(*,*,1;LMAX-1)+C6*
     1   V(*,*,1;LMAX-1))*RR(*,*,1;LMAX-1)
      DU(5,4) = ((C3-C0)*W +C5*U +C6*V )*RR(*,*,2;LMAX)
      D(5,5) = C0*RR(*,*,1;LMAX-1)
      DU(5,5) = C0*RR(*,*,2;LMAX)
      DO 31 N = 2,5
      DO 31 M = 1,5
      DEFINE(D1,D(N,M)),(DU1,DU(N,M))
      A(N,M)=A(N,M)+DRE*D1(*,*,2;LMAX-1)
      B(N,M)=B(N,M)-DRE*(D1(*,*,3;LMAX)+DU1(*,*,2;LMAX-1))
      C(N,M)=C(N,M)+DRE*DU1(*,*,3;LMAX)
   31 CONTINUE
      RETURN
      END
      SUBROUTINE VISRHS
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH,
     1 DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ,
     2 RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2  INT3
```

```
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(243,30,6,30)
      COMMON/VAR0/S(24,30,6,30)
      COMMON/VAR1/X(243,30,30),Y(243,30,30),Z(243,30,30)
      COMMON/VAR3/P(120,30),XX(4),YY(4),ZZ(4)
      LEVEL 2,Q,S,X,Y,Z
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F(5)
      COMMON/COUNT/NC,NC1
      DYNAMIC S0,S1,S2,S3,S4,S5,S6,T1,T2,T3,T4,T5,T6,T7
      DYNAMIC RJ,ZZ,U,V,W,E,T8,T9,T10,T11,T12,T13,T14,T15,T16
      DYNAMIC DU,DV,DW,DEI,F2,F3,F4,F5,R2,R3,R4,R5,S2,F
      COMMON TURMU(720,30)
      IF(LAMIN.EQ.1) CALL MUTUR
      GKPR = GAMMA/PR
      PRTR = PR/0.9
      DRE = MD/(RE*DZ1**2)
      TURM = TURMU(K,L,J)
      VNU = RMUE+TURM
      GKAP = RMUE+PRTR*TURM
      DO 2000 L=1,LMAX
      U1(1:KMAX-2,1:JSL,L)=U(*,L,*)
      V1(1:KMAX-2,1:JSL,L)=V(*,L,*)
      W1(1:KMAX-2,1:JSL,L)=W(*,L,*)
2000  CONTINUE
      DO 1000,N=1,3
      DEFINE(Z1,ZZ1(N)),(ZZZ,ZZ(N))
      Z1(1:KMAX-2,1:JSL,L)=ZZZ(*,L,*)
      S0 = (ZZ1(1)**2+ZZ1(2)**2+ZZ1(3)**2)*RJ1
      S1 = (S0 +ZZ1(1)**2/3.*RJ1)*VNU
      S2 = (S0 +ZZ1(2)**2/3.*RJ1)*VNU
      S3 = (S0 +ZZ1(3)**2/3.*RJ1)*VNU
      S4 = (ZZ1(1)*ZZ1(2)/3.*RJ1)*VNU
      S5 = (ZZ1(1)*ZZ1(3)/3.*RJ1)*VNU
      S6 = (ZZ1(2)*ZZ1(3)/3.*RJ1)*VNU
      S0 = S0 *GKPR*GKAP
      E  = Q5*RR1-.5*(U **2+V **2+W **2)
      T1 = S1(*,*,2)+S1(*,*,1)
      T2 = S2(*,*,2)+S2(*,*,1)
      T3 = S3(*,*,2)+S3(*,*,1)
      T4 = S4(*,*,2)+S4(*,*,1)
      T5 = S5(*,*,2)+S5(*,*,1)
      T6 = S6(*,*,2)+S6(*,*,1)
      T7 = U1(*,*,2)*S1(*,*,2)+U1(*,*,1)*S1(*,*,1)
      T8 = V1(*,*,2)*S2(*,*,2)+V1(*,*,1)*S2(*,*,1)
      T9 = W1(*,*,2)*S3(*,*,2)+W1(*,*,1)*S3(*,*,1)
      T10= U1(*,*,2)*S4(*,*,2)+U1(*,*,1)*S4(*,*,1)
      T11= U1(*,*,2)*S5(*,*,2)+U1(*,*,1)*S5(*,*,1)
      T12= V1(*,*,2)*S4(*,*,2)+V1(*,*,1)*S4(*,*,1)
      T13= V1(*,*,2)*S6(*,*,2)+V1(*,*,1)*S6(*,*,1)
      T14= W1(*,*,2)*S5(*,*,2)+W1(*,*,1)*S5(*,*,1)
      T15= W1(*,*,2)*S6(*,*,2)+W1(*,*,1)*S6(*,*,1)
      T16= S0(*,*,2)+S0(*,*,1)
      DU = (U1(*,*,2)-U1(*,*,1))
      DV = (V1(*,*,2)-V1(*,*,1))
      DW = (W1(*,*,2)-W1(*,*,1))
      DEI = (E(*,*,2)-E(*,*,1))
      R2 = T1*DU+T4*DV+T5*DW
      R3 = T4*DU+T2*DV+T6*DW
      R4 = T5*DU+T6*DV+T3*DW
      R5 = (T7+T12+T14)*DU+(T8+T10+T15)*DV+(T9+T11+T13)*DW+T16*DEI
      DO 1000,L=2,LMAX-1
```

```
      T1 = S1(*,*,L+1)+S1(*,*,L)
      T2 = S2(*,*,L+1)+S2(*,*,L)
      T3 = S3(*,*,L+1)+S3(*,*,L)
      T4 = S4(*,*,L+1)+S4(*,*,L)
      T5 = S5(*,*,L+1)+S5(*,*,L)
      T6 = S6(*,*,L+1)+S6(*,*,L)
      T7 = U1(*,*,L+1)*S1(*,*,L+1)+U1(*,*,L)*S1(*,*,L)
      T8 = V1(*,*,L+1)*S2(*,*,L+1)+V1(*,*,L)*S2(*,*,L)
      T9 = W1(*,*,L+1)*S3(*,*,L+1)+W1(*,*,L)*S3(*,*,L)
      T10= U1(*,*,L+1)*S4(*,*,L+1)+U1(*,*,L)*S4(*,*,L)
      T11= U1(*,*,L+1)*S5(*,*,L+1)+U1(*,*,L)*S5(*,*,L)
      T12= V1(*,*,L+1)*S4(*,*,L+1)+V1(*,*,L)*S4(*,*,L)
      T13= V1(*,*,L+1)*S6(*,*,L+1)+V1(*,*,L)*S6(*,*,L)
      T14= W1(*,*,L+1)*S5(*,*,L+1)+W1(*,*,L)*S5(*,*,L)
      T15= W1(*,*,L+1)*S6(*,*,L+1)+W1(*,*,L)*S6(*,*,L)
      T16= S0(*,*,L+1)+S0(*,*,L)
      DU = (U1(*,*,L+1)-U1(*,*,L))
      DV = (V(*,*,L+1)-V(*,*,L))
      DW = (W(*,*,L+1)-W(*,*,L))
      DEI = (E(*,*,L+1)-E(*,*,L))
      F2 = T1*DU+T4*DV+T5*DW
      F3 = T4*DU+T2*DV+T6*DW
      F4 = T5*DU+T6*DV+T3*DW
      F5 = (T7+T12+T14)*DU+(T8+T10+T15)*DV+(T9+T11+T13)*DW+T16*DEI
      DEFINE (F(1),(1;KMAX-2,1;JSL,1;LMAX-2))
      F(1) = 0.
      F(2) = F2-R2
      F(3) = F3-R3
      F(4) = F4-R4
      F(5) = F5-R5
      R2 = F2
      R3 = F3
      R4 = F4
      R5 = F5
      DO 40 N = 1,5
40    S2(N)=S2(N)+F(N)*DRE
1000  CONTINUE
      RETURN
      END
```

APPENDIX D

IMPLICIT PSEUDO COMPILATION

```
      SUBROUTINE STEP
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH
     1  ,DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ
     2  ,RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3,LSL,JSL
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(24,30,6,30)
      COMMON/VAR0/S(24,30,5,30)
      COMMON/VAR1/X(24,30,30),Y(24,30,30),Z(24,30,30)
      COMMON /VAR3/P(120,30),XX(4),YY(4),ZZ(4)
      COMMON/COUNT/NC,NC1
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F
      LEVEL 2 Q,S,X,Y,Z
      DYNAMIC A,B,C,D,S,SD,XX(4),YY(4),ZZ(4),XKL ,YKL
      DYNAMIC F,F1(5),F2,S1(5)
      DYNAMIC ZKL
      DYNAMIC XJL ,YJL ,ZJL ,XKJ ,YKJ ,ZKJ
      DYNAMIC QT1,QT2,QT3,QT4,QT5,TV
      DYNAMIC RJ,RR,U,V,W,UU,UT,C1,C2,C3,C4,C5,C6,C7
      DYNAMIC RMJ,RF,XK,YK,ZK,XL,YL,ZL,XJ,YJ,ZJ
      DYNAMIC DPLUS(5,5),DMIN(5,5),QDES(8,2),XYZ(3,2)
      DYNAMIC Q1,Q2,Q3,Q4,Q5
      DYNAMIC X1,Y1,Z1
      INTEGER QINPOS,QOUTPOS,XYZPOS,FDESC(5),SDESC(5)
C
      CALL BC
      CALL RHS
      CALL SMOOTH
C
C  COMPUTE L2 RESIDUAL
C
C   THE CONDITIONAL BRANCH CODE
C***         IF(NC.EQ.1) GO TO 5
C***         IF(NC-(NC/10)*10)5,5,6
C***       5 CONTINUE
CAN BE REPLACED BY

      IF(MOD(NC,10).NE.0)GO TO 6

      RESID = 0.0
      KMH = KM
      LMH = LM
      DO 10 N = 1,5
      DO 10 L = 1,LMH
      DO 10 K = 1,KMH
      DO 10 J=2,JM
  10  RESID = RESID+S(K,L,N,J)**2
      RESID = RESID/((JM-1)*(KMH-1)*(LMH-1))
      RESID = SQRT(RESID)/(DT*.00005)
C
C CAN BE VECTORIZED AUTOMATICALLY,HOWEVER THE THROUGHPUT OF THIS LOCAL
C  LOOP WILL BE LIMITED TO THE BANDWIDTH OF THE INTERMEDIATE STORAGE
      WRITE(6,100) NC,RESTD
  100 FORMAT(1H0,3HN= ,I5,3X,13HL2 RESIDUAL= ,F16.8)
    6 CONTINUE
C
C
      RM = SMU
      C8 = 1.+2.*RM
      GAM2 = 2.-GAMMA
C   THE NESTED DO LOOP:
C***     DO 20 L = 2,LM
```

```
C***      DO 20 K = 2,KM
C    CAN BE VECTORIZED IN SEGMENTS AS FOLLOWS:
C
C
C***FILTRX
C
      JA=2
      JB=JMAX-1
C#   MAP RJ       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=1
C#   MAP XKL      GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX;  WK=2
C#   MAP YKL      GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX;  WK=3
C#   MAP ZKL      GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX;  WK=4
C#   MAP Q1       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=5
C#   MAP Q2       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=6
C#   MAP Q3       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=7
C#   MAP Q4       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=8
C#   MAP Q5       GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX;  WK=9
      DO 1000 L=2,LMAX,LSL
C    LSL IS THE NUMBER OF SLICES(COLUMNS) IN L PER PROCESSING PASS
      LSM=LSL-1
C
      DO 5 N=1,5
      DEFINE (F1(N),F(2:KMAX-1,L:L+LSM,N,2:JMAX-1))
 5    DEFINE (S1(N),S(2:KMAX-1,L:L+LSM,N,2:JMAX-1))
      Q1=Q(*,L:L+LSM,1,2:JMAX-1)
      Q2=Q(*,L:L+LSM,2,2:JMAX-1)
      Q3=Q(*,L:L+LSM,3,2:JMAX-1)
      Q4=Q(*,L:L+LSM,4,2:JMAX-1)
      Q5=Q(*,L:L+LSM,5,2:JMAX-1)
C#   MAP Q1'      GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX
C#   MAP Q2'      GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX
C#   MAP Q3'      GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX
C#   MAP Q4'      GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX
C#   MAP Q5'      GTHR:IM  NR=JMAX*LSL,RS=KMAX-2,ST=KMAX:KMAX*LMAX
C
      RJ=Q(2:KMAX-1,L:L+LSM,6,*)
C
C#   MAP RJ'      GTHR:IM  NR=JMAX,RS=KMAX*LSL,ST=KMAX*LMAX
C
      XKL=X(*,L-1:L+LSM+1,2:JMAX-1)
C
C#   MAP XKL'     GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX
C
      YKL=Y(*,L-1:L+LSM+1,2:JMAX-1)
C
C#   MAP YKL'     GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX
C
      ZKL=Z(*,L-1:L+LSM+1,2:JMAX-1)
C
C#   MAP ZKL'     GTHR:IM  NR=JMAX-2,RS=KMAX*(LSL+2),ST=KMAX*LMAX
C
      XK=(XKL(3:KMAX,2:LSL+1,*)-XKL(1:KMAX-2,2:LSL+1,*))*DY2
C
C#   VEC XK       SUB:MUL    VL=KMAX*LSL*(JMAX-2); WK=11,RK=2
C#   MAP XK       CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2);
C*                           WK=21,RK=11
C
      YK=(YKL(3:KMAX,2:LSL+1,*)-YKL(1:KMAX-2,2:LSL+1,*))*DY2
C
C#   VEC YK       SUB:MUL    VL=KMAX*LSL*(JMAX-2); WK=12,RK=3
C#   MAP YK       CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2);
C*                           WK=22,RK=12
C
```

```
      ZK=(ZKL(3:KMAX,2:LSL+1,*)-ZKL(1:KMAX-2,2:LSL+1,*))*DY2            000990
C                                                                       000991
C*   VEC ZK      SUB:MUL      VL=KMAX,LSL*(JMAX-2): WK=13,RK=4          000992
C*   MAP ZK      CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2): 000993
C*                            WK=23,RK=13                               000994
C                                                                       000995
      XL=(XKL(*,3:LSL,*)-XKL(*,1:LSL-2,*))*DZ2                          001000
C                                                                       001001
C*   VEC XL      SUB:MUL      VL=KMAX*LSL*(JMAX-2): WK=14               001002
C*   MAP XL      CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2): 001003
C*                            RK=14,WK=24                               001004
C                                                                       001005
      YL=(YKL(*,3:LSL,*)-YKL(*,1:LSL-2,*))*DZ2                          001010
C                                                                       001011
C*   VEC YL      SUB:MUL      VL=KMAX*LSL*(JMAX-2): WK=15               001012
C*   MAP YL      CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2): 001013
C*                            RK=15,WK=25                               001014
C                                                                       001015
      ZL=(ZKL(*,3:LSL,*)-ZKL(*,1:LSL-2,*))*DZ2                          001020
C                                                                       001021
C*   VEC ZL      SUB:MUL      VL=KMAX*LSL*(JMAX-2): WK=16               001022
C*   MAP ZL      CMPS:MM  CVL=KMAX*LSL*(JMAX-2),RL=(KMAX-2)*LSL*(JMAX-2): 001023
C*                            RK=16,WK=26                               001024
C                                                                       001025
      XX(1)=(YK*ZL-ZK*YL)*RJ(*,*,2:JMAX-1)                              001030
C                                                                       001031
C*   VEC T1      MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2): RK=26            001032
C                                                                       001033
      XX(2)=(ZK*XL-XK*ZL)*RJ(*,*,2:JMAX-1)                              001040
C                                                                       001041
C*   VEC T2      MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001042
C*   VEC XX1     MUL          VL=(KMAX-2)*LSL*(JMAX-2)  <RJ*T1>         001043
C*   SSS XX2     MUL                                    <RJ*T2>         001044
C                                                                       001045
      XX(3)=(XK*YL-YK*XL)*RJ(*,*,2:JMAX-1)                              001050
C                                                                       001051
C*   VEC T1      MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001052
C                                                                       001055
      XX(4)=-OMEGA*(ZKL(2:KMAX-1,1:LSL,2:JMAX-1)*XX(2)                  001060
     1    -YKL(2:KMAX-1,1:LSL,2:JMAX-1)*XX(3))                          001061
C                                                                       001062
C*   VEC T2      MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001063
C*   VEC XX3     MUL          VL=(KMAX-2)*LSL*(JMAX-2)                  001064
C*   SSS XX4     MUL                                    <OM*T2>         001065
C                                                                       001066
      D(1,2)  =XX(1)*HDX                                                001070
      D(1,3)  =XX(2)*HDX                                                001080
C                                                                       001081
C*   VEC D12     MUL          VL=(KMAX-2)*LSL*(JMAX-2)                  001082
C*   SSS D13     MUL                                                    001083
C                                                                       001084
      D(1,4)  =XX(3)*HDX                                                001090
      D(1,1)  =XX(4)*HDX                                                001100
C                                                                       001101
C*   VEC D14     MUL          VL=(KMAX-2)*LSL*(JMAX-2)                  001102
C*   SSS D11     MUL                                                    001103
C                                                                       001110
C*******AMATRX                                                          001120
C                                                                       001130
      RR=1/Q1                                                           001140
C                                                                       001141
C*   VEC RR      DIV          VL=(KMAX-2)*LSL*(JMAX+2): RK=5            001142
C                                                                       00114³
```

```
      U=RR*Q2                                                           001150
      V=RR*Q3                                                           001160
C                                                                       001161
C*   VEC U       MUL; RK=7   VL=(KMAX-2)*LSL*(JMAX-2)                   001162
C*   $$$ V       MUL                                                    001163
C                                                                       001164
      W=RR*Q4                                                           001170
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)                                   001180
C                                                                       001181
C*   VEC T1      MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)  <U*D12+V*D13>    001182
C*   VEC UU      MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)  <T1+W*(RR*Q4)>   001183
C*                           RK=8                                       001184
C                                                                       001185
      UT = U**2+V**2+W**2                                               001190
C                                                                       001191
C*   VEC T1      MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)  <U*U+V*V>        001192
C*   VEC UT      MUL:ADD     VL=(KMAX-2)*LSL*(JMAX-2)  <T1+W*W>         001193
C                                                                       001194
      C1 = GAMI*UT*.5                                                   001200
C                                                                       001201
C*   VEC C1      MUL*MUL     VL=(KMAX-2)*LSL*(JMAX-2)                   001202
C                                                                       001203
      C2=RR*GAMMA*Q5                                                    001210
C                                                                       001211
C*   VEC C2      MUL:MUL     VL=(KMAX-2)*LSL*(JMAX-2)                   001212
C                                                                       001213
      C3=C2-C1                                                          001220
      C4=D(1,1)*UU                                                      001230
C                                                                       001231
C*   VEC C3      SUB        'VL=(KMAX-2)*LSL*(JMAX-2)                   001232
C*   $$$ C4      ADD                                                    001233
C                                                                       001234
      C5=GAMI*U                                                         001240
      C6=GAMI*V                                                         001250
C                                                                       001251
C*   VEC C5      MUL         VL=(KMAX-2)*LSL*(JMAX-2)                   001252
C*   $$$ C6      MUL                                                    001253
C                                                                       001254
      C7=GAMI*W                                                         001260
      DEFINE (D(1,5),(1:KMAX-2,1:LSL-2,1:JMAX-2))                       001270
      D(1,5)=0                                                          001280
C                                                                       001281
C*   VEC C7      MUL         VL=(KMAX-2)*LSL*(JMAX-2)                   001282
C*   $$$ D15     ADD(0)                                                 001283
C                                                                       001284
      D(2,1)  = D(1,2)*C1-U*UU                                          001290
C                                                                       001291
C*   VEC D21     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001292
C                                                                       001293
      D(2,2)  = C4+D(1,2)*GAM2*U                                        001300
C                                                                       001301
C*   VEC D22     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001302
C                                                                       001303
      D(2,3)  = -D(1,2)*C6+D(1,3)*U                                     001310
C                                                                       001311
C*   VEC D23     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001312
C                                                                       001313
      D(2,4)  = -D(1,2)*C7+D(1,4)*U                                     001320
C                                                                       001321
C*   VEC D24     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001322
C                                                                       001323
      D(2,5)  = D(1,2)*GAMI                                             001330
      D(3,1)  = D(1,3)*C1-V*UU                                          001340
```

```
C                                                                       001341
C*   VEC D31     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001342
C                                                                       001343
      D(3,2) = D(1,2)*V-D(1,3)*C5                                       001350
C                                                                       001351
C*   VEC D32     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001352
C                                                                       001353
      D(3,3) = C4+D(1,3)*GAM2*V                                         001360
C                                                                       001361
C*   VEC D33     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001362
C                                                                       001363
      D(3,4) = -D(1,3)*C7+D(1,4)*V                                      001370
C                                                                       001371
C*   VEC D34     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001372
C                                                                       001373
      D(3,5) = D(1,3)*GAMI                                              001380
C                                                                       001381
C*   VEC D25     MUL         VL=(KMAX-2)*LSL*(JMAX-2)                   001382
C*   $$$ D35     MUL                                                    001383
C                                                                       001384
      D(4,1) = D(1,4)*C1-W*UU                                           001390
C                                                                       001391
C*   VEC D41     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001392
C                                                                       001393
      D(4,2) = D(1,2)*W-D(1,4)*C5                                       001400
C                                                                       001401
C*   VEC D42     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001402
C                                                                       001403
      D(4,3) = D(1,3)*W-D(1,4)*C6                                       001410
C                                                                       001411
C*   VEC D43     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001412
C                                                                       001413
      D(4,4) = C4+D(1,4)*GAM2*W                                         001420
C                                                                       001421
C*   VEC D44     MUL:MUL:ADD VL=(KMAX-2)*LSL*(JMAX-2)                   001422
C                                                                       001423
      D(4,5) = D(1,4)*GAMI                                              001430
C                                                                       001431
C*   VEC D45     MUL         VL=(KMAX-2)*LSL*(JMAX-2)                   001432
C                                                                       001433
      D(5,1) = (-C2+2.*C1)*UU                                           001440
C                                                                       001441
C*   VEC D51     MUL:ADD:MUL VL=(KMAX-2)*LSL*(JMAX-2)                   001442
C                                                                       001443
      D(5,2) = D(1,2)*C3-C5*UU                                          001450
C                                                                       001451
C*   VEC D52     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001452
C                                                                       001453
      D(5,3) = D(1,3)*C3-C6*UU                                          001460
C                                                                       001461
C*   VEC D53     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001462
C                                                                       001463
      D(5,4) = D(1,4)*C3-C7*UU                                          001470
C                                                                       001471
C*   VEC D54     MUL:MUL:SUB VL=(KMAX-2)*LSL*(JMAX-2)                   001472
C                                                                       001473
      D(5,5) = D(1,1)+GAMMA*UU                                          001480
C                                                                       001481
C*   VEC D55     MUL:ADD     VL=(KMAX-2)*LSL*(JMAX-2)                   001482
C                                                                       001483
C                                                                       001490
C******END OF AMATRX                                                    001500
C                                                                       001510
```

```
      RMJ=RM/RJ(*,*,2:JMAX-1)
C
C#   VEC RMJ     DIV          VL=(KMAX-2)*LSL*(JMAX-2)
C
      RR=RMJ*RJ(*,*,1:JMAX-2)
      RF=RMJ*RJ(*,*,3:JMAX)
C
C#   VEC RR      MUL          VL=(KMAX-2)*LSL*(JMAX-2)
C*   SSS RF      MUL
C
      DO 23 N=1,5
      DO 22 M=1,5
      DEFINE (B(N,M),(1:KMAX-2,1:LSM,1:JMAX-2)
      DEFINE (D1,D(N,M))
      A(N,M)=-D1(*,*,1:JMAX-2)
      C(N,M)=D1(*,*,3:JMAX)
C
C#   VEC ANM     NOP          VL=(KMAX-2)*LSL*(JMAX-2)
C*   SSS CNM     NOP
C
 22   B(N,M)=0
C
C#   MAP BNM     SCTR:MM  NR=1,RS=(KMAX-2)*LSL*(JMAX-2)
C
      A(N,N) = A(N,N)-RR
      B(N,N) = CB
C
C#   MAP BNM     STCR:MM  NR=1,RS=(KMAX-2)*LSL*(JMAX-2)
C
      C(N,N) = C(N,N)-RF
C
C#   VEC ANN     SUB          VL=(KMAX-2)*LSL*(JMAX-2)
C*   SSS CNN     SUB
C
 23    F1(N)=S1(N)
C
C#   MAP F1N     GTHR:IM  NR=JMAX-2,RS=(KMAX-2)*LSL,ST=KMAX*LMAX
C
C
C*****END OF FILTRX
C
C    S MUST BE ZERO ON B.C.
C
C#   MAP RJT     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=1
C#   MAP XJLT    GTHR:II      NR=KMAX*JSL*LMAX,ST=KMAX:1: WK=2
C#   MAP YJLT    GTHR:II      NR=KMAX*JSL*LMAX,ST=KMAX:1: WK=3
C#   MAP ZJLT    GTHR:II      NR=KMAX*JSL*LMAX,ST=KMAX:1: WK=4
C#   MAP Q1T     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=5
C#   MAP Q2T     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=6
C#   MAP Q3T     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=7
C#   MAP Q4T     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=8
C#   MAP Q5T     GTHR:II      NR=KMAX*JSL*(LMAX-2),ST=KMAX:1: WK=9
C
      CALL BTRI(2,JM)
      DO 24 N=1,5
      S1(N)=F1(N)
C
C#   MAP S1N     SCTR:MI  NR=JMAX-2,RS=(KMAX-2)*LSL,ST=KMAX*LMAX
C
 24   CONTINUE
C
C
 1000 CONTINUE
```

ORIGINAL PAGE IS
OF POOR QUALITY

```
C
C*******FILTRY
C
      KA = 2
      KB = KMAX-1
C#   MAP RJ      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=1
C#   MAP XJL     GTHR;IM  RS=KMAX*JSL*LMAX; WK=2
C#   MAP YJL     GTHR;IM  RS=KMAX*JSL*LMAX; WK=3
C#   MAP ZJL     GTHR;IM  RS=KMAX*JSL*LMAX; WK=4
C#   MAP Q1      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=5
C#   MAP Q2      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=6
C#   MAP Q3      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=7
C#   MAP Q4      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=8
C#   MAP Q5      GTHR;IM  RS=KMAX*JSL*(LMAX-2); WK=9
      JSM = JSL-1
      DO 2000 J=2,JMAX,JSL
      DO 6 N=1,5
      DEFINE (F1(N),F(2;LMAX-1,J;J+JSM,N,2;KMAX-1))
  6   DEFINE (S1(N),F(2;LMAX-1,J;J+JSM,N,2;KMAX-1))
C
      RJ = RJT(2;LMAX-1,J;J+JSM,2;KMAX)
C
C#   MAP RJ'     GTHR;IM  RS=KMAX*JSL*(LMAX-2)
C#   MAP XJL'    GTHR;IM RS=KMAX*JSL*LMAX
C#   MAP YJL'    GTHR;IM RS=KMAX*JSL*LMAX
C#   MAP ZJL'    GTHR;IM RS=KMAX*JSL*LMAX
C
      XJL = XJT
      YJL = YJLT
      ZJL = ZJLT
      Q1 = Q1T
C
C#   MAP Q1'     GTHR;IM  RS=KMAX*JSL*LMAX
C
      Q2 = Q2T
C
C#   MAP Q2'     GTHR;IM  RS=KMAX*JSL*LMAX
C
      Q3 = Q3T
C
C#   MAP Q3'     GTHR;IM  RS=KMAX*JSL*LMAX
C
      Q4 = Q4T
C
C#-  MAP Q4'     GTHR;IM  RS=KMAX*JSL*LMAX
C
      Q5 = Q5T
C
C#   MAP Q5'     GTHR;IM RS=KMAX*JSL*LMAX
C
C
C
      DO 9 N=1,5
      DO 9 K=1,KMAX
      DEFINE (F2,F1(N))
 9    F2(*,*,K)=S(K,2;LMAX-1,N,J;J+JSM)
C  THIS ACCOMPLISHES THE MOVE OF S TO THE F ARRAY
C
C
      XJ=(XJL(2;LMAX-1,3;JSL+2,2;KMAX-1)-XJL(2;LMAX-1,1;JSL,2;KMAX-1))
     1  *DX2
C#-  VEC XJ      SUB;MUL     VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP XJ   CMPS;MM  CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)
```

```
C
      YJ=(YJL(2:LMAX-1,3:JSL+2,2:KMAX-1)-XJL(2:LMAX-1,1:JSL,2:KMAX-1))
     1   *DX2
C#   VEC YJ      SUB:MUL      VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP YJ    CMPS:MM   CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)
C
      ZJ=(ZJL(2:LMAX-1,3:JSL+2,2:KMAX-1)-XJL(2:LMAX-1,1:JSL,2:KMAX-1))
     1   *DY2
C#   VEC ZJ      SUB:MUL      VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP ZJ    CMPS:MM   CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)
C
      XL=(XJL(3:LMAX,2:JSL+1,2:KB)-XJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2
C#   VEC XL      SUB:MUL      VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP XL    CMPS:MM   CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)
C
      YL=(YJL(3:LMAX,2:JSL+1,2:KB)-YJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2
C#   VEC YL      SUB:MUL      VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP YL    CMPS:MM   CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)

      ZL=(ZJL(3:LMAX,2:JSL+1,2:KB)-ZJL(1:LMAX-2,2:JSL+1,2:KB))*DZ2
C#   VEC ZL      SUB:MUL      VL=LMAX*(JSL+2)*(KMAX-2)
C#   MAP ZL    CMPS:MM   CVL=LMAX*(JSL+2)*(KMAX-2),VL=(LMAX-2)*JSL*(KMAX-2)
C
      YY(1)=(ZJ*YL-YJ*ZL)*RJ(*,*,2:KMAX-1)
      YY(2)=(XJ*ZL-XL*ZJ)*RJ(*,*,2:KMAX-1)
      YY(3)=(YJ*XL-XJ*YL)*RJ(*,*,2:KMAX-1)
      YY(4)=-OMEGA*(ZJL(2:LMAX-1,2:JSM,2:KMAX-1)*YY(2)
     1                YJL(2:LMAX-1,2:JSM,2:KMAX-1)*YY(3))
      D(1,2) =YY(1)*HDY
      D(1,3) =YY(2)*HDY
      D(1,4) =YY(3)*HDY
      D(1,1) =YY(4)*HDY
C
C******AMATRX
      RR=1/Q1
      U=RR*Q2
      V=RR*Q3
      W=RR*Q4
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)
      UT = U**2+V**2+W**2
      C1 = GAMI*UT*.5
      C2=RR*GAMMA*Q5
      C3=C2-C1
      C4=D(1,1)+UU
      C5=GAMI*U
      C6=GAMI*V
      C7=GAMI*W
      DEFINE (D(1,5),(1:LMAX-2,1:JSL,1:KMAX-2))
      D(1,5) = 0.
      D(2,1) = D(1,2)*C1-U*UU
      D(2,2) = C4+D(1,2)*GAM2*U
      D(2,3) = -D(1,2)*C6+D(1,3)*U
      D(2,4) = -D(1,2)*C7+D(1,4)*U
      D(2,5) = D(1,2)*GAMI
      D(3,1) = D(1,3)*C1-V*UU
      D(3,2) = D(1,2)*V-D(1,3)*C5
      D(3,3) = C4+D(1,3)*GAM2*V
      D(3,4) = -D(1,3)*C7+D(1,4)*V
      D(3,5) = D(1,3)*GAMI
      D(4,1) = D(1,4)*C1-W*UU
      D(4,2) = D(1,2)*W-D(1,4)*C5
      D(4,3) = D(1,3)*W-D(1,4)*C6
      D(4,4) = C4+D(1,4)*GAM2*W
```

```
      D(4,5)  = D(1,4)*GAMI
      D(5,1)  = (-C2+2.*C1)*UU
      D(5,2)  = D(1,2)*C3-C5*UU
      D(5,3)  = D(1,3)*C3-C6*UU
      D(5,4)  = D(1,4)*C3-C7*UU
      D(5,5)  = D(1,1)+GAMMA*UU
C
C******END OF AMATRX
C
C     RJ IS RETAINED FROM THE DIFFERENCING CALCULATION
      RMJ=RM*RJ(*,*,2:KMAX-1)
      RR=RMJ*RJ(*,*,1:KMAX-2)
      RF=RMJ*RJ(*,*,3:KMAX)
      DO 34 N=1,5
      DO 33 M=1,5
      DEFINE (D1,D(N,M))
      DEFINE(B(N,M),(1:LMAX-2,1:JMAX-2,1:KMAX-2))
C
C   NOTE THAT B HAS CHANGED SHAPE AS WE ARE DEALING WITH
C      THE TRANSPOSED MESH FOR THIS SWEEP
      A(N,M)=-D1(*,*,1:KMAX-2)
      B(N,M)=0
 33   C(N,M)=D1(*,*,3:KMAX)
      A(N,N)  = A(N,N)-RR
      B(N,N)  = C8
      C(N,N)  = C(N,N)-RF
 34   CONTINUE
C    F HAS BEEN PREMOVED AT TOP OF LOOP
C
C********END OF FILTRY
C
      CALL BTRI(2,KM)
      DO 31 N=1,5
      DEFINE (S2,S1(N)),(F2,F1(N))
      DO 31 K=2,KB
      S(K,2:LMAX-1,N,J:J+JSM) = F2(2:LMAX-1,1:JSL-2,K)
 31   CONTINUE
C
C
C
 2000 CONTINUE
C******FILTRZ
      DEFINE (RJ,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q1,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q2,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q3,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q4,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (Q5,(1:KMAX,1:JSL,1:LMAX))
      DEFINE (XKJ,(1:KMAX,1:JSL+2,1:LMAX))
      DEFINE (YKL,(1:KMAX,1:JSL+2,1:LMAX))
      DEFINE (YKL,(1:KMAX,1:JSL+2,1:LMAX))
      LA =2
      LB =LMAX-1
      JSM = JSL-1
      DO 3000 J=2,JMAX,JSL
C
      DO 61 N=1,5
   61 DEFINE (F1(N),(F(2:KMAX-1,J:J+JSM,N,2:LMAX-1))
      DO 81 L=1,LMAX
      RJ(1:KMAX-2,1:JSL,L)=Q(2:KMAX-1,L,6,J:J+JSM)
      XKJ(1:KMAX,1:JSL+2,L)=X(1:KMAX,L,J-1:J+JSL)
      YKJ(1:KMAX,1:JSL+2,L)=Y(1:KMAX,L,J-1:J+JSL)
      ZKJ(1:KMAX,1:JSL+2,L)=Z(1:KMAX,L,J-1:J+JSL)
```

```
      Q1(1;KMAX,1;JSL,L)=Q(2;KMAX-1,L,1,J;J+JSM)
      Q2(1;KMAX,1;JSL,L)=Q(2;KMAX-1,L,2,J;J+JSM)
      Q3(1;KMAX,1;JSL,L)=Q(2;KMAX-1,L,3,J;J+JSM)
      Q4(1;KMAX,1;JSL,L)=Q(2;KMAX-1,L,4,J;J+JSM)
      Q5(1;KMAX,1;JSL,L)=Q(2;KMAX-1,L,5,J;J+JSM)
C
C
 81   CONTINUE
      DO 91 N=1,5
      DO 91 L=1,LMAX
      DEFINE (F2,F1(N))
 91   F2(*,*,L)=S(2;KMAX-1,L,N,J;J+JSM)
      XK=(XKJ(3;KMAX,2;JSL+1,2;LB)-XKJ(1;KMAX-2,2;JSL+1,2;LB))*DY2
      YK=(YKJ(3;KMAX,2;JSL+1,2;LB)-YKJ(1;KMAX-2,2;JSL+1,2;LB))*DY2
      ZK=(ZKJ(3;KMAX,2;JSL+1,2;LB)-ZKJ(1;KMAX-2,2;JSL+1,2;LB))*DY2
      XJ=(XKJ(2;KMAX-1,3;JSL+2,2;LB)-XKJ(2;KMAX-1,1;JSL,2;LB))*DX2
      YJ=(YKJ(2;KMAX-1,3;JSL+2,2;LB)-YKJ(2;KMAX-1,1;JSL,2;LB))*DX2
      ZJ=(ZKJ(2;KMAX-1,3;JSL+2,2;LB)-ZKJ(2;KMAX-1,1;JSL,2;LB))*DX2
      ZZ(1)=(YJ*ZK-ZJ*YK)*RJ
      ZZ(2)=(XK*ZJ-XJ*ZK)*RJ
      ZZ(3)=(XJ*YK-YJ*XK)*RJ
      ZZ(4)=-OMEGA*(ZKJ(2;KMAX-1,2;JSM,2;LB)*ZZ(2)
     1               -YKJ(2;KMAX-1,2;JSM,2;LB)*ZZ(3))

      D(1,2)  =ZZ(1)*HDZ
      D(1,3)  =ZZ(2)*HDZ
      D(1,4)  =ZZ(3)*HDZ
      D(1,1)  =ZZ(4)*HDZ
C
C******AMATRX
      RR=1./Q1
      U=RR*Q2
      V=RR*Q3
      W=RR*Q4
      UU = U*D(1,2)+V*D(1,3)+W*D(1,4)
      UT = U**2+V**2+W**2
      C1 = GAMI*UT*.5
      C2=RR*Q5*GAMMA
      C3=C2-C1
      C4=D(1,1)+UU
      C5=GAMI*U
      C6=GAMI*V
      C7=GAMI*W
      DEFINE(D(1,5),(1;KMAX-2,1;JSL,1;LMAX-2))
      D(1,5)  = 0.
      D(2,1)  = D(1,2)*C1-U*UU
      D(2,2)  = C4+D(1,2)*GAM2*U
      D(2,3)  = -D(1,2)*C6+D(1,3)*U
      D(2,4)  = -D(1,2)*C7+D(1,4)*U
      D(2,5)  = D(1,2)*GAMI
      D(3,1)  = D(1,3)*C1-V*UU
      D(3,2)  = D(1,2)*V-D(1,3)*C5
      D(3,3)  = C4+D(1,3)*GAM2*V
      D(3,4)  = -D(1,3)*C7+D(1,4)*V
      D(3,5)  = D(1,3)*GAMI
      D(4,1)  = D(1,4)*C1-W*UU
      D(4,2)  = D(1,2)*W-D(1,4)*C5
      D(4,3)  = D(1,3)*W-D(1,4)*C6
      D(4,4)  = C4+D(1,4)*GAM2*W
      D(4,5)  = D(1,4)*GAMI
      D(5,1)  = (-C2+2.*C1)*UU
      D(5,3)  = D(1,3)*C3-C6*UU
      D(5,4)  = D(1,4)*C3-C7*UU
```

```
      D(5,5) = D(1,1)+GAMMA*UU
C
C*****END OF ANATRX
C
C     RJ IS RETAINED FROM DIFFERENCING OPERATION
      RMJ = RM/RJ(*,*,2:LMAX-1)
      RF = RMJ*RJ(*,*,3:LMAX)
      RR = RMJ*RJ(*,*,1:LMAX-2)
      DO 43 N=1,5
      DO 42 M=1,5
      DEFINE(D1,D(N,M))
      DEFINE(B(N,M),(1:KMAX-2,1:JSL,1:LMAX-2))
      A(N,M)=-D1(*,*,1:LMAX-2)
      B(N,M)=0
 42   C(N,M)=D1(*,*,3:LMAX)
      A(N,N) = A(N,N)-RR
      B(N,N) = C8
      C(N,N) = C(N,N)-RF
 43   CONTINUE
C
C*******END OF FILTRZ
C
      IF(INVISC.EQ.1) CALL VISMAT(J,K)
      CALL BTRI(2,LM)
      Q1=Q1+F(1)
      Q2=Q2+F(2)
      Q3=Q3+F(3)
      Q4=Q4+F(4)
      Q5=Q5+F(5)
      DO 44 L=2,LMAX-1
      Q(2:KMAX-1,L,1,J:J+JSM)=Q1(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,2,J:J+JSM)=Q2(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,3,J:J+JSM)=Q3(1:KMAX-2,1:JSL-2,L)
      Q(2:KMAX-1,L,4,J:J+JSM)=Q4(1:KMAX-2,1:JSL-2,L)
 44   Q(2:KMAX-1,L,5,J:J+JSM)=Q5(1:KMAX-2,1:JSL-2,L)
 3000 CONTINUE
      RETURN
      END
      SUBROUTINE BTRI(ILA,IUA)
      COMMON/BTRID/A(5,5),B(5,5),C(5,5),D(5,5),F(5)
      DYNAMIC H(5,5),A,B,C,D,F
      DYNAMIC L11,L21,L22,L31,L32,L33,L41,L42,L43,L44
      DYNAMIC L51,L52,L53,L54,L55
      DYNAMIC U11,U21,U22,U31,U32,U33,U41,U42,U43,U44
      DYNAMIC U51,U52,U53,U54,U55
      DYNAMIC D1,D2,D3,D4,D5
      DYNAMIC U12,U13,U14,U15,U23,U24,U25,U34,U35,U45
      DYNAMIC A1(5,5),B1(5,5),C1(5,5),F1(5)
      REAL L11,L21,L22,L31,L32,L33,L41,L42,L43,L44,L51,L52,L53,L54,L55
      IL=ILA
      IU=IUA
      IS=IL+1
      IE=IU-1
      DO 1 N=1,5
      DEFINE (F1(N),F(N)(*,*,1))
      DO 1 M=1,5
      DEFINE (B1(N,M),B(N,M)(*,*,1))
 1    DEFINE (C1(N,M),C(N,M)(*,*,1))
C     INSERT LUDEC
      L11=1./B (1,1)
C
C#  VEC L11     DIV          VL=SSL*SMAX
C
```

```
C   SSL=NUMBER OF PLANES IN SLAB                                        004334
C     SMAXX= EITHER LMAX,KMAX OR JMAX DEPENDING ON SWEEP DIRECTION      004335
      L21=B1(2,1)                                                       004340
C                                                                       004341
C#  MAP L21     GTHR:MM  NR=1,RS=SSL*SMAX; WK=1                         004342
C                                                                       004343
      U12=B1(1,2)*L11                                                   004350
      L22=1./(B1(2,2)-L21*U12)                                          004360
C                                                                       004361
C#  VEC U12     MUL          VL=SSL*SSMAX; RK=1                         004362
C*  $$$ T1      MUL:SUB                                                 004363
C#  VEC L22     DIV          VL=SSL*SSMAX                               004364
C                                                                       004365
      U13=B1(1,3)*L11                                                   004370
      U14 = B1(1,4)*L11                                                 004380
C                                                                       004381
C#  VEC U13     MUL          VL=SSL*SSMAX                               004382
C*  $$$ U14     MUL                                                     004383
C                                                                       004384
      U15=B1(1,5)*L11                                                   004390
C                                                                       004391
C#  VEC U15     MUL          VL=SSL*SSMAX                               004392
C*  $$$ L31     MUL*(1)                                                 004393
C                                                                       004394
      L31=B1(3,1)                                                       004400
C                                                                       004401
      L32=B1(3,2)-L31*U12                                               004410
C                                                                       004411
C#  VEC L32     MUL:SUB      VL=SSL*SSMAX                               004412
C                                                                       004413
      U23=(B1(2,3)-L21*U13)*L22                                         004420
C                                                                       004421
C#  VEC U23     MUL:SUB:MUL VL=SSL*SSMAX                                004422
C                                                                       004423
      L33=1./(B1(3,3)-U13*L31-U23*L32)                                  004430
C                                                                       004431
C#  VEC T1      MUL:MUL:ADD VL=SSL*SSMAX                                004432
C#  VEC T1      SUB          VL=SSL*SSMAX                               004433
C#  VEC U24     DIV          VL=SSL*SSMAX                               004434
C                                                                       004435
      U24=(B1(2,4)-L21*U14)*L22                                         004440
C                                                                       004441
C#  VEC U24     MUL:SUB:MUL VL=SSL*SSMAX                                004442
C                                                                       004443
      U25=(B1(2,5)-L21*U15)*L22                                         004450
C                                                                       004451
C#  VEC U25     MUL:SUB:MUL VL=SSL*SSMAX                                004452
C                                                                       004453
      L41=B1(4,1)                                                       004460
C                                                                       004461
C#  MAP L41     GTHR:MM  NR=1,RL=SSL*SSMAX; WK=1                        004462
C                                                                       004463
      L42=B1(4,2)-L41*U12                                               004470
      L43=B1(4,3)-L41*U13-L42*U23                                       004480
C                                                                       004481
C#  VEC L42     MUL:SUB      VL=SSL*SSMAX; RK=1                         004482
C*  $$$ T1      MUL                                                     004483
C#  VEC L43     MUL:SUB:SUB VL=SSL*SSMAX                                004484
C                                                                       004485
      U34=(B1(3,4)-L31*U14-L32*U24)*L33                                 004490
C                                                                       004491
C#  VEC T1      MUL:MUL:ADD VL=SSL*SSMAX                                004492
C#  VEC U34     SUB:MUL      VL=SSL*SSMAX                               004493
```

```
C
      L44=1./(B1(4,4)-U14*L41-U24*L42-U34*L43)
C
C#   VEC T1       MUL;MUL;ADD VL=SSL*SSMAX
C#   VEC T1       MUL;SUB;SUB VL=SSL*SSMAX
C#   VEC L44      DIV         VL=SSL*SSMAX
C
      U35=(B1(3,5)-L31*U15-L32*U25)*L33
C
C#   VEC T1       MUL;MUL;ADD VL=SSL*SSMAX
C#   VEC U35      SUB;MUL     VL=SSL*SSMAX
C
      L51=B1(5,1)
C
C#   MAP L51      GTHR;MM  NR=1,RS=SSL*SSMAX; WK=1
C
      L52=B1(5,2)-L51*U12
      L53=B1(5,3)-L51*U13-L52*U23
C
C#   VEC L52      MUL;SUB     VL=SSL*SSMAX
C*   SSS T1       MUL
C*   VEC L53      MUL;SUB;SUB VL=SSL*SSMAX; RK=1
C
      L54=B1(5,4)-L51*U14-L52*U24-L53*U34
C
C#   VEC T1       MUL;MUL;ADD VL=SSL*SSMAX
C#   VEC L54      MUL;SUB;SUB VL=SSL*SSMAX
C
      U45=(B1(4,5)-L41*U15-L42*U25-L43*U35)*L44
C
C#   VEC T1       MUL;MUL;ADD VL=SSL*SSMAX   <L42*U25-L43*U35>
C#   VEC T1       MUL;ADD;SUB VL=SSL*SSMAX   <-L41*U15+B45-T1>
C
      L55=1./(B1(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C
C#   VEC U45      MUL         VL=SSL*SSMAX
C*   SSS T1       MUL;SUB     VL=SSL*SSMAX   <B55-L54*U45>
C#   VEC T2       MUL;MUL;ADD VL=SSL*SSMAX   <L51*U15+L52*U25>
C#   VEC T1       MUL;SUB;SUB VL=SSL*SSMAX   <T1-L53*U35-T2>
C#   VEC L55      DIV         VL=SSL*SSMAX
C
C     COMPUTE LITTLE R S
      D1=L11*F1(1)
      D2=L22*(F1(2)-L21*D1)
C
C#   VEC D1       MUL         VL=SSL*SSMAX
C#   VEC T1       MUL;SUB     VL=SSL*SSMAX   <F1-L21*D1>
C
      D3=L33*(F1(3)-L31*D1-L32*D2)
C
C#   VEC D2       MUL         VL=SSL*SSMAX   <T1*L22>
C*   SSS T1       MUL;SUB     VL=SSL*SSMAX   <F13-L32*D2>
C#   VEC D3       MUL;SUB;MUL VL=SSL*SSMAX   <L33*(T1-L31*D1)>
C
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
C
C#   VEC T1       MUL;MUL;ADD VL=SSL*SSMAX   <L42*D2+L43*D3>
C#   VEC T1       MUL;SUB;SUB VL=SSL*SSMAX   <F14-T1-L41*D1>
C
C
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C
C#   VEC D4       MUL         VL=SSL*SSMAX   <L44*T1>
```

```
C*   SSS T1      MUL:SUB          <F15-L54*D4>                                004633
C*   VEC T2      MUL:MUL:ADD VL=SSL*SSMAX   <L51*D1+L52*D2>                   004634
C*   VEC T1      MUL:SUB:SUB VL=SSL*SSMAX   <T1-T2-L53*D3>                    004635
C                                                                              004636
C      COMPUTE BIG R S                                                         004640
       F1(5)=05                                                                004650
C                                                                              004651
C*   VEC D5      MUL         VL=SSL*SSMAX                                      004652
C*   SSS F15     MUL*(1)                                                       004653
C                                                                              004654
       F1(4)=D4-U45*05                                                         004660
       F1(3)=D3-U34*F1(4)-U35*D5                                               004670
C                                                                              004671
C*   VEC F14     MUL:SUB     VL=SSL*SSMAX                                      004672
C*   SSS T1      MUL                    <U34*F14>                              004673
C*   VEC F13     MUL:SUB:SUB VL=SSL*SSMAX   <D3-U35*D5-T1>                     004674
C*   VEC L55     DIV         VL=SSL*SSMAX                                      004676
       F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5                                     004680
C                                                                              004681
C*   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX   <U24*F14+U25*D5>                  004682
C*   VEC F12     MUL:SUB:SUB VL=SSL*SSMAX   <D2-U23*F13-T1>                    004683
C                                                                              004684
       F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5                           006690
C                                                                              004691
C*   VEC T1      MUL:MUL:ADD VL=SSL*SSMAX                                      004692
C*   VEC T2      MUL:MUL:ADD VL=SSL*SSMAX                                      004693
C*   VEC T2      SUB:SUB     VL=SSL*SSMAX                                      004694
C                                                                              004695
C      COMPUTE C PRIME FOR FIRST ROW                                           004700
       DO 12 M=1,5                                                             004710
       D1=L11*C1(1,M)                                                          004720
       D2=L22*(C1(2,M)-L21*D1)                                                 004730
C                                                                              004731
C*   VEC D1      MUL         VL=SSL*SSMAX                                      004732
C*   SSS T1      MUL:SUB                                                       004733
C                                                                              004734
       D3=L33*(C1(3,M)-L31*D1-L32*D2)                                          004740
C                                                                              004741
C*   VEC D2      MUL         VL=SSL*SSMAX                                      004742
C*   SSS T1      MUL:SUB                                                       004743
C*   VEC D3      MUL:SUB:MUL VL=SSL*SSMAX                                      004744
C                                                                              004745
       D4=L44*(C1(4,M)-L41*D1-L42*D2-L43*D3)                                   004750
       D5=L55*(C1(5,M)-L51*D1-L52*D2-L53*D3-L54*D4)                            006760
C                                                                              004761
C*   VEC T1      MUL:MUL:ADD VL=SSL*SSMAX   <L42*D2+L43*D3>                    004762
C*   VEC T1      MUL:SUB:SUB VL=SSL*SSMAX   <C14M-T1-L41*D1>                   004763
C*   VEC D4      MUL         VL=SSL*SSMAX   <L44*T1>                           004764
C*   SSS T1      MUL:SUB                         <C15M-L54*D4>                 004765
C*   VEC T1      MUL:MUL:ADD VL=SSL*SSMAX   <L52*D2+L53*D3>                    004766
C*   VEC T1      MUL:ADD:ADD VL=SSL*SSMAX   <L51*D1+T1+T2>                     004767
C                                                                              004768
       B1(5,M)=D5                                                              004770
C                                                                              004771
C*   VEC D5      MUL         VL=SSL*SSMAX   <L55*T1>                           004772
C*   SSS B15M    MUL                                                           004773
C                                                                              004774
       B1(4,M)=D4-U45*D5                                                       004780
       B1(3,M) = D3-U34*B1(4,M)-U35*D5                                         004790
C                                                                              004791
C*   VEC B14M    MUL:SUB     VL=SSL*SSMAX                                      004792
C*   SSS T1      MUL                      <U34*B14M>                           004793
C*   VEC B13M    MUL:SUB:SUB VL=SSL*SSMAX   <D3-T1-U35*D5>                     004794
```

```
C
      B1(2,M) = D2-U23*B1(3,M)-U24*B1(4,M)-U25*D5
C
C#  VEC T1      MUL:MUL:ADD VL=SSL*SSMAX
C#  VEC B2M     MUL:SUB:SUB VL=SSL*SSMAX
C
12    B1(1,M) = D1-U12*B1(2,M)-U13*B1(3,M)-U14*B1(4,M)-U15*D5
C
C#  VEC T1      MUL:MUL:ADD VL=SSL*SSMAX
C#  VEC T2      MUL:MUL:ADD VL=SSL*SSMAX
C#  VEC B11M    SUB:SUB     VL=SSL*SSMAX
C
C      COMPUTE B PRIME*BIGR
      DO 13 I=IS,IE
      DO 2 N=1,5
      DEFINE (F1(N),F(N)(*,*,I));(F2(N),F(N)(*,*,I-1))
      DO 2 M=1,5
      DEFINE (A1(N,M),A(N,M)(*,*,I))
      DEFINE (B1(N,M),B(N,M)(*,*,I)),(B2(N,M),B(N,M)(*,*,I-1))
 2    DEFINE (C1(N,M),C(N,M)(*,*,I))
      DO 14 N=1,5
 14   F1(N)=F1(N)-A1(N,1)*F2(1)-A1(N,2)*F2(2)-A1(N,3)*F2(3)
     *         -A1(N,4)*F2(4)-A1(N,5)*F2(5)
C
C#    VEC T1     MUL:MUL:ADD  VL=SSL*SSMAX
C#    VEC T2     MUL:MUL:ADD  VL=SSL*SSMAX
C#    VEC T1     MUL:MUL:ADD  VL=SSL*SSMAX
C#    VEC F1N    SUB          VL=SSL*SSMAX
C
C      COMPUTE B PRIME
      DO 11 N=1,5
      DO 11 M=1,5
 11   H(N,M)=B1(N,M)-A1(N,1)*B2(1,M)-A1(N,2)*B2(2,M)-A1(N,3)*
     *         B2(3,M)-A1(N,4)*B2(4,M)-A1(N,5)*B2(5,M)
C
C#    VEC T1     MUL:MUL:ADD  VL=SSL*SSMAX
C#    VEC T2     MUL:MUL:ADD  VL=SSL*SSMAX
C#    VEC T1     MUL:ADD:ADD  VL=SSL*SSMAX
C#    VEC HNM    SUB          VL=SSL*SSMAX
C
C      INSERT LUDEC AGAIN
      L11=1./H(1,1)
C
C#    VEC L11    DIV          VL=SSL*SSMAX
C
      L21=H(2,1)
C
C#    MAP L21    GTHR:MM  NR=1,RS=SSL*SSMAX
C
      U12=H(1,2)*L11
C
C#    VEC U12    MUL          VL=SSL*SSMAX
C*    $$$ T1     MUL:SUB      VL=SSL*SSMAX
C#    VEC L22    DIV          VL=SSL*SSMAX
C
      L22=1./(H(2,2)-L21*U12)
      U13=H(1,3)*L11
      U14=H(1,4)*L11
C
C#  VEC U13     MUL           VL=SSL*SSMAX
C*  $$$ U14     MUL
C
      U15=H(1,5)*L11
```

```
      L31=H(3,1)                                                          005060
C                                                                         005061
C*   MAP L31     GTHR:MM  NR=1,RS=SSL*SSMAX                               005062
C                                                                         005063
      L32=H(3,2)-L31*U12                                                  005070
      U23=(H(2,3)-L21*U13)*L22                                            005080
C                                                                         005081
C*   VEC L32     MUL:SUB       VL=SSL*SSMAX                               005082
C*   SSS T1      MUL                          <U23*L32>                   005083
C*   VEC U23     MUL:SUB:MUL   VL=SSL*SSMAX                               005084
C                                                                         005085
      L33=1./(H(3,3)-U13*L31-U23*L32)                                     005090
C                                                                         005091
C*   VEC T1      MUL:SUB:SUB   VL=SSL*SSMAX                               005092
C*   VEC L33     DIV           VL=SSL*SSMAX                               005093
C                                                                         005094
      U24=(H(2,4)-L21*U14)*L22                                            005100
C                                                                         005101
C*   VEC U24     MUL:SUB:MUL   VL=SSL*SSMAX                               005102
C                                                                         005103
      U25=(H(2,5)-L21*U15)*L22                                            005110
C                                                                         005111
C*   VEC U15     MUL           VL=SSL*SSMAX   <H15*L11>                   005112
C*   SSS T9      MUL:SUB                                                  005113
C                                                                         005114
      L41=H(4,1)                                                          005120
C                                                                         005121
C*   MAP L41     GTHR:MM  NR=1,RS=SSL*SSMAX                               005122
C                                                                         005123
      L42=H(4,2)-L41*U12                                                  005130
C                                                                         005131
C*   VEC L42     MUL:SUB       VL=SSL*SSMAX                               005132
C*   SSS T1      MUL                            <L42*U23>                 005133
C                                                                         005134
      L43=H(4,3)-L41*U13-L42*U23                                          005140
C                                                                         005141
C*   VEC L43     MUL:SUB:SUB   VL=SSL*SSMAX   <H43-T1-L41*U13>            005142
C                                                                         005143
      U34=(H(3,4)-L31*U14-L32*U24)*L33                                    005150
C                                                                         005151
C*   VEC T1      MUL:MUL:ADD   VL=SSL*SSMAX  <L31*U14+L32*U24>            005152
C*   VEC U34     SUB:MUL       VL=SSL*SSMAX   <(H34-T1)*L33>              005153
C*   SSS T1      MUL                          <U34*L43>                   005154
C                                                                         005155
      L44=1./(H(4,4)-U14*L41-U24*L42-U34*L43)                             005160
C                                                                         005161
C*   VEC T2      MUL:MUL:ADD   VL=SSL*SSMAX   <U14*L41+U24*L42>           005162
C*   VEC T1      SUB:SUB       VL=SSL*SSMAX   <H44-T1-T2>                 005163
C*   VEC L44     DIV           VL=SSL*SSMAX                               005164
C                                                                         005165
      U35=(H(3,5)-L31*U15-L32*U25)*L33                                    005170
C*   VEC U25     MUL           VL=SSL*SSMAX   <T9*L22>                    005171
C*   SSS T1      MUL:SUB       VL=SSL*SSMAX   <H35-L32*U25>               005172
C*   VEC U35     MUL:SUB:MUL   VL=SSL*SSMAX   <(T1-L31*U15)*L33>          005173
C                                                                         005174
      L51=H(5,1)                                                          005180
C                                                                         005181
C*   MAP L51     GTHR:MM   NR=1,RS=SSL*SSMAX                              005182
C                                                                         005183
      L52=H(5,2)-L51*U12                                                  005190
      L53=H(5,3)-L51*U13-L52*U23                                          005200
C                                                                         005201
C*   VEC L52     MUL:SUB       VL=SSL*SSMAX                               005202
```

```
C*   SSS T1      MUL
C#   VEC L53     MUL:SUB:SUB  VL=SSL*SSMAX
C
      L54=H(5,4)-L51*U14-L52*U24-L53*U34
C
C#   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C#   VEC L54     MUL:SUB:SUB  VL=SSL*SSMAX
C
      U45=(H(4,5)-L41*U15-L42*U25-L43*U35)*L44
C
C#   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C#   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX
C
      L55=1./(H(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C
C#   VEC U45     MUL          VL=SSL*SSMAX
C*   SSS T1      MUL:SUB                     <H55-L54*U45>
C#   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX
C#   VEC L55     DIV          VL=SSL*SSMAX
C#   VEC L55     DIV         VL=SSL*SSMAX
C
C     COMPUTE LITTLE RIS
      D1=L11*F1(1)
C
C#   VEC D1      MUL          VL=SSL*SSMAX
C*   SSS T1      MUL:SUB      VL=SSL*SSMAX   <F12-L21*D1>
C
      D2=L22*(F1(2)-L21*D1)
C#   VEC D2      MUL          VL=SSL*SSMAX   <L22*T1>
C*   SSS T1      MUL:SUB                     <F13-L32*D2>
C
      D3=L33*(F1(3)-L31*D1-L32*D2)
C
C#   VEC D3      MUL:SUB:MUL  VL=SSL*SSMAX   <L33*(T1-L31*D1)>
C
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C
C#   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX   <L42*D2+L43*D3>
C#   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX   <F1-L41*D1-T1>
C#   VEC D4      MUL          VL=SSL*SSMAX   <L44*T1>
C*   SSS T1      MUL:SUB      VL=SSL*SSMAX   <F15-L54*D4>
C#   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX   <L52*D2+L53*D3>
C#   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX   <T1-L51*D1-T2>
C
C     COMPUTE BIG RIS
      F1(5)=D5
      F1(4)=D4-U45*D5
C
C#   VEC D5      MUL          VL=SSL*SSMAX   <L55*T1>
C*   SSS F15     MUL                         <L55*D5>
C#   VEC F14     MUL:SUB      VL=SSL*SSMAX
C*   SSS T3      MUL                         <F14*U34>
C
      F1(3)=D3-U34*F1(4)-U35*D5
      F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5
C
C#   VEC F13     MUL:SUB:SUB  VL=SSL*SSMAX   <D3-T3-U35*D5>
C#   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C#   VEC F12     MUL:SUB:SUB  VL=SSL*SSMAX
C
      F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5
C
```

```
C#   VEC T1       MUL:MUL:ADD  VL=SSL*SSMAX
C#   VEC T2       MUL:MUL:ADD  VL=SSL*SSMAX
C*   SSS T1       MUL:SUB                              <C12M-L21*D1>
C
C      COMPUTE C PRIMES
       DO 15 M=1,5
       D1=L11*C1(1,M)
       D2=L22*(C1(2,M)-L21*D1)
C
C#   VEC D1       MUL          VL=SSL*SSMAX
C*   SSS T1       MUL:SUB                         <C12-L21*D1>
C
       D3=L33*(C1(3,M)-L31*D1-L32*D2)
C
C#   VEC D2       MUL          VL=SSL*SSMAX   <L22*T1>
C*   SSS T1       MUL:SUB                      <C13M-L32*D2>
C#   VEC D3       MUL:SUB:MUL  VL=SSL*SSMAX   <L33*(T1-L31*D1>
C
       D4=L44*(C1(4,M)-L41*D1-L42*D2-L43*D3)
C
C#   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX
C
       D5=L55*(C1(5,M)-L51*D1-L52*D2-L53*D3-L54*D4)
C
C#   VEC T1       MUL:SUB:SUB VL=SSL*SSMAX
C#   VEC D4       MUL         VL=SSL*SSMAX
C*   SSS T1       MUL:SUB                         <C15-L54*D4>
C#   VEC T2       MUL:MUL:SUB VL=SSL*SSMAX
C#   VEC T1       MUL:SUB:SUB VL=SSL*SSMAX
C
       B1(5,M)=D5
       B1(4,M)=D4-U45*D5
C
C#   VEC D5       MUL         VL=SSL*SSMAX
C*   SSS B14      MUL:SUB     VL=SSL*SSMAX
C#   MAP D5       GTHR:MM     NR=1,RS=SSL*SSMAX
C
       B1(3,M) = D3-U34*B1(4,M)-U35*D5
       B1(2,M) = D2-U23*B1(3,M)-U24*B1(4,M)-U25*D5
C
C#   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX
C#   VEC B13      SUB         VL=SSL*SSMAX
C*   SSS T1       MUL:SUB
C#   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX
15     B1(1,M) = D1-U12*B1(2,M)-U13*B1(3,M)-U14*B1(4,M)-U15*D5
C#   VEC B12      SUB         VL=SSL*SSMAX
C*   SSS T1       MUL:SUB
C#   VEC T2       MUL:MUL:ADD VL=SSL*SSMAX
C#   VEC B11      MUL:SUB:SUB VL=SSL*SSMAX
C
13     CONTINUE
       I=IU
       DO 3 N=1,5
       DEFINE (F1(N),F(N)(*,*,IU)),(F2(N),F(N)(*,*,IU-1))
       DO 3 M=1,5
       DEFINE (A1(N,M),A(N,M)(*,*,IU))
 3     DEFINE (B1(N,M),B(N,M)(*,*,IU)), (B2(N,M),B(N,M)(*,*,IU-1))
C       COMPUTE B PRIME*BIG R FOR LAST ROW
       DO 17 N=1,5
 17    F1(N)=F1(N)-A1(N,1)*F2(1)-A1(N,2)*F2(2)-A1(N,3)*
     1        F2(3)-A1(N,4)*F2(4)-A1(N,5)*F2(5)
C
C#   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX
```

```
C#   VEC T2  '    MUL:MUL:ADD VL=SSL*SSMAX
C#   VEC T1       SUM:SUB:SUB VL=SSL*SSMAX
C#   VEC F1       SUB         VL=SSL*SSMAX
C
C       COMPUTE B PRIME
      DO 18 N=1,5
      DO 18 M=1,5
 18   H(N,M)=B1(N,M)-A1(N,1)*B2(1,M)-A1(N,2)*B2(2,M)-A1(N,3)*
     *B2(3,M)-A1(N,4)*B2(4,M)-A1(N,5)*B2(5,M)
C
C#   VEC T1       MUL:MUL:ADD VL=SSL*SSMAX
C#   VEC T2       MUL:MUL:ADD VL=SSL*SSMAX
C#   VEC T1       MUL:SUB:SUB VL=SSL*SSMAX
C#   VEC HNM      SUB         VL=SSL*SSMAX
C
C      INSERT LUDEC AGAIN
      L11=1./H(1,1)
      L21=H(2,1)
C
C#   MAP L21      GTHR:MM     NR=1,RS=SSL*SSMAX
C#   VEC L11      DIV         VL=SSL*SSMAX
C
      U12=H(1,2)*L11
      L22=1./(H(2,2)-L21*U12)
C
C#   VEC U12      MUL         VL=SSL*SSMAX
C*   SSS T1       MUL:SUB
C#   VEC L22      DIV         VL=SSL*SSMAX
C
      U13=H(1,3)*L11
      U14=H(1,4)*L11
C
C#   VEC U13      MUL           VL=SSL*SSMAX
C*   SSS U14      MUL           VL=SSL*SSMAX
C
      U15=H(1,5)*L11
      L31=H(3,1)
      L32=H(3,2)-L31*U12
      U23=(H(2,3)-L21*U13)*L22
C#   MAP L31      GTHR:MM   NR=1,RS=SSL*SSMAX
C#   VEC U23      MUL:SUB:MUL  VL=SSL*SSMAX
C#   VEC L32      MUL:SUB      VL=SSL*SSMAX
C*   SSS T1       MUL                              <U23*L32>
C
      L33=1./(H(3,3)-U13*L31-U23*L32)
C
C#   VEC T1       MUL:SUB:SUB  VL=SSL*SSMAX
C#   VEC L33      DIV          VL=SSL*SSMAX
C
      U24=(H(2,4)-L21*U14)*L22
C
C#   VEC U24      MUL:SUB:MUL  VL=SSL*SSMAX
C
      U25=(H(2,5)-L21*U15)*L22
C
C#   VEC U15      MUL          VL=SSL*SSMAX
C*   SSS T9       MUL:SUB
C
      L41=H(4,1)
C
C#   MAP L41      GTHR:MM   NR=1,RS=SSL*SSMAX
C
      L42=H(4,2)-L41*U12
```

```
C
C#  VEC L42     MUL:SUB        VL=SSL*SSMAX
C*  SSS T1      MUL                                <L42*U23>
C
      L43=H(4,3)-L41*U13-L42*U23
C
C#  VEC L43     MUL:SUB:SUB    VL=SSL*SSMAX
C
      U34=(H(3,4)-L31*U14-L32*U24)*L33
C
C#  VEC T1      MUL:MUL:ADD    VL=SSL*SSMAX
C#  VEC U34     SUB:MUL        VL=SSL*SSMAX
C*  SSS T1      MUL                                <U34*L43>
C
      L44=1./(H(4,4)-U14*L41-U24*L42-U34*L43)
C
C#  VEC T2      MUL:MUL:ADD    VL=SSL*SSMAX
C#  VEC T1      MUL:SUB:SUB    VL=SSL*SSMAX
C#  VEC L44     DIV            VL=SSL*SSMAX
      U35=(H(3,5)-L31*U15-L32*U25)*L33
C
C#  VEC U25     MUL            VL=SSL*SSMAX  <T9*L22>
C*  SSS T1      MUL:SUB                            <L32*U25>
C#  VEC U35     MUL:SUB:MUL    VL=SSL*SSMAX
C
      L51=H(5,1)
C
C#  MAP L51     GTHR:MM        NR=1,RS=SSL*SSMAX
C
      L52=H(5,2)-L51*U12
      L53=H(5,3)-L51*U13-L52*U23
      L54=H(5,4)-L51*U14-L52*U24-L53*U34
C
C#  VEC L52     MUL:SUB        VL=SSL*SSMAX
C*  SSS T1      MUL
C#  VEC L53     MUL:SUB:SUB    VL=SSL*SSMAX
C#  VEC T1      MUL:MUL:ADD    VL=SSL*SSMAX
C#  VEC L54     MUL:SUB:SUB    VL=SSL*SSMAX
C
      U45=(H(4,5)-L41*U15-L42*U25-L43*U35)*L44
C
C#  VEC T1      MUL:MUL:ADD    VL=SSL*SSMAX
C#  VEC T1      MUL:SUB:SUB    VL=SSL*SSMAX
C
      L55=1./(H(5,5)-L51*U15-L52*U25-L53*U35-L54*U45)
C
C#  VEC U45     MUL            VL=SSL*SSMAX
C*  SSS T1      MUL:SUB
C#  VEC T2      MUL:MUL:ADD    VL=SSL*SSMAX
C#  VEC T1      MUL:SUB:SUB    VL=SSL*SSMAX
C#  VEC L55     DIV            VL=SSL*SSMAX
C
C     COMPUTE LITTLE RIS
      D1=L11*F1(1)
C
C#  VEC D1      MUL            VL=SSL*SSMAX
C*  SSS T1      MUL:SUB                            <F13-L31*D1>
C
      D2=L22*(F1(2)-L21*D1)
C
C#  VEC D2      MUL            VL=SSL*SSMAX
C*  SSS T1      MUL:SUB
C
```

```
      D3=L33*(F1(3)-L31*D1-L32*D2)
C
C*   VEC D3      MUL:SUB:MUL  VL=SSL*SSMAX
C
      D4=L44*(F1(4)-L41*D1-L42*D2-L43*D3)
      D5=L55*(F1(5)-L51*D1-L52*D2-L53*D3-L54*D4)
C
C*   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX
C*   VEC D4      MUL          VL=SSL*SSMAX
C*   $$$ T1      MUL:SUB
C*   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX
C
C     COMPUTE BIG R1S
      F1(5)=D5
      F1(4)=D4-U45*D5
C
C*   VEC D5      MUL          VL=SSL*SSMAX
C*   $$$ F1      MUL:SUB
C*   MAP F15     GTHR:MM      NR=1,RS=SSL*SSMAX
C
      F1(3)=D3-U34*F1(4)-U35*D5
      F1(2)=D2-U23*F1(3)-U24*F1(4)-U25*D5
C
C*   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC F13     SUB          VL=SSL*SSMAX
C*   $$$ T1      MUL:SUB
C*   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX
C
      F1(1)=D1-U12*F1(2)-U13*F1(3)-U14*F1(4)-U15*D5
C
C*   VEC F12     SUB          VL=SSL*SSMAX
C*   $$$ T1.     MUL:SUB
C*   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC F11     MUL:SUB:SUB  VL=SSL*SSMAX
C
      I=IU
20    I=I-1
      DO 4 N=1,5
      DEFINE (F1(N),F(N)(*,*,I)),(F2(N),F(N)(*,*,I+1))
    4 CONTINUE
      DO 19 N=1,5
 19   F1(N)=F1(N)-F2(1)*B1(N,1)-F2(2)*B1(N,2)-F2(3)*B1(N,3)
     *       -F2(4)*B1(N,4)-F2(5)*B1(N,5)
C
C*   VEC T1      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC T2      MUL:MUL:ADD  VL=SSL*SSMAX
C*   VEC T1      MUL:SUB:SUB  VL=SSL*SSMAX
C*   VEC F1N     SUB          VL=SSL*SSMAX
      IF (I.GT.IL)GOTO20
      RETURN
      END
```

Appendix E

LOADER CONVENTIONS

This section contains formats for loader tables, some of which can provide information that could be useful for debugging. The following are the loader tables that the system uses during error processing:

Module Header Table

Code Block Table

External/Entry Table

Debug Symbol Table

Symbol Definition Table

Pseudo Address Vector Table

Error processing information is provided for every object module loaded to produce a controllee file, including object modules of user-specified files and required object modules for system library files. Figure E-1 is a dump of a typical controllee file, illustrating the error processing information area at the end of the dumped file. A pointer to the error processing information is placed in register #D. The register contains the total word length of error information in its upper 16 bits and the starting address in its lower 48 bits.

GENERAL TABLE STRUCTURE

The loader works with files that are composed of one or more object modules. Each object module consists of a number of standard tables; each table begins with a standard two-word header:

1	ASCII Table Name 64	
2	Length 16	Address 48

Word 1	Name of the table in ASCII	
Word 2	Length	Length of the table in full words
	Address	Bit difference between first word of the respective table and word 1 of module header table; i.e.: Back pointer (bits) + address of first word of respective table (bits) = address of word 1 of header table (bits)

```
January 1, 1981  11:10:09
    10000000  0000 0000 4000 00A0   0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0000        @
               *** ZERO ***
    10002200  0000 0000 0000 00A0   0000 0000 0200 0080   0000 0000 0000 0100   0000 0000 0000 0000
               *** ZERO ***
    10002C00  0000 0000 0000 0000   0000 0000 4000 0000   0000 0000 0000 0000   0000 0000 0000 0000              @
               *** ZERO ***
    10003600  0000 0000 6002 0001   0000 0002 2002 0002   0000 0000 0000 0000   0000 0000 0000 0000
               *** ZERO ***
    10008500  0000 0000 0000 0A00   0000 0000 0000 0580   0000 0000 0000 0001   0000 0000 0000 0000
    10008600  0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0000   0006 0000 4004 0000                           @
    10008700  0006 0000 4004 0000   0000 0000 0000 0000   0065 0000 4001 0000   0000 0000 4000 0080        @          @      @
               *** ZERO ***
    10010000  4D41 494E 2E20 2020   03AC 0000 4003 8800   7B1E C010 7A1A 001C   BE11 0000 4000 E640   MAIN.     @            @ @
    10010100  3610 0011 3E23 0000   7F24 0023 7F25 0023   7F26 0023 7F27 0023   3E28 0048 7F29 0028   6   >#   $ # % # & # ' #>( H ) (
    10010200  3E23 0040 7F2A 0023   3E28 0001 7F2B 002A   9E23 FFFF 0202 0014   3E2C 0013 7F2D 2C23   ># @ * #>(   + ( #       >,   -,#
    10010300  7A00 2E2F 3E30 0014   7F2D 302F BE31 0004   0100 0013 7F2D 0031   3E32 0001 7F2D 3233    ./>0   -0/ 1        - 1>2   -23
    10010400  7B00 2E34 3833 0034   7F2D 3234 3E35 0001   7F36 0035 7F2D 2037   7F37 00FE 0900 0000    .483 4 -24>5   6 5  -7 7
    10010500  00EE 00FE 3E38 0080   7F2D 3838 3E39 0002   1238 393A 3E38 00FF   3E3A 0001 8004 003A      >8   -88>9   89:>8  >:     :
   10028000  2020 2020 2020 2028   2020 2020 2020 2029   2020 2020 2020 202C   2829 2C00 0000 0000          (        )       ,(),
   10028100  0000 0000 0000 0048   0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0000         H
   10028200  0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0001   5041 5241 4D45 5445                           PARAMETE
   10028300  5220 4F52 2046 4F52   4D41 5420 4552 524F   5220 464F 554E 442E   2020 5554 494C 4954   R OR FORMAT ERROR FOUND.  UTILIT
   10028400  5920 5445 524D 494E   4154 4544 2E00 0000   554E 4142 4C45 2054   4F20 434F 4D50 4C45   Y TERMINATED.  UNABLE TO COMPLE

   10028500  5445 2055 5449 4C49   5459 2E20 5452 5920   4147 4149 4E2F 0000   0000 0000 0000 0000   TE UTILITY. TRY AGAIN.
   10028600  0002 0000 0000 0000   5350 4543 4946 5920   5041 5241 4D45 5445   5253 2D00 0000 0000           SPECIFY PARAMETERS-
   10028700  2020 2028 4649 4C45   312C 4649 4C45 322C   4C3D 4C4E 472C 413D   4144 5231 2C42 3D41      (FILE1,FILE2,L=LNG,A=ADR1,B=A
   10028800  4452 322C 4E3D 4552   524C 494D 4954 2900   5553 4552 2053 5045   4349 4649 4544 204C   DR2,N=ERRLIMIT) USER SPECIFIED L
   10028900  4E47 2049 5320 4C4F   4E47 4552 2054 4841   4E20 4649 4C45 204C   4E47 2E20 5452 554E   NG IS LONGER THAN FILE LNG. TRUN
   10028A00  4341 5445 4420 544F   2030 3030 3030 3030   0000 0000 0000 0040   5448 4953 2046 494C   CATED TO 0000000       @THIS FIL
   10028B00  4520 434F 554C 4420   4E4F 5420 4245 204F   5045 4E45 4420 2D00   2020 2020 2020 2020   E COULD NOT BE OPENED -
   10028C00  2020 2020 2020 2020   2020 2020 2020 2020   2020 2020 2020 2020   434F 4D50 4152 4520                          COMPARE
   10028D00  5445 524D 494E 4154   4544 2E00 0000 0000   4649 4C45 4E41 4D45   2020 2020 2020 2020   TERMINATED.     FILENAME
   10028E00  2020 434F 554C 4420   4E4F 5420 4245 204D   4150 5045 442D 494E   2E20 434F 4D50 4152     COULD NOT BE MAPPED-IN. COMPAR
   10028F00  4520 5445 524D 494E   4154 4544 2E00 0000   2020 2020 2020 2020   2020 2041 4E44 0000   E TERMINATED.                AND
   10029000  2020 2020 2020 2020   2043 4F4D 5041 5245   4420 4551 5541 4C4C   592E 2055 5449 4C49             COMPARED EQUALLY. UTILI
   10029100  5459 2044 4F4E 452E   0000 0000 0000 0000   0000 0000 0000 0000   0000 0000 0000 0000   TY DONE.

                                     ERROR PROCESSING INFORMATION AREA

              *** ZERO ***
   10048800  204D 4F44 554C 4520   0000 0000 0000 0000   4D41 494E 2E20 2020   594D 4448 4D53 2020     MODULE          MAIN.   YMDHMS
   10048900  0016 2020 204C 4C4C   03AC 0000 0000 1940   0001 0000 4000 0000   0002 0000 4003 5A40        LLL        @   @        @  @
   10048A00  0301 0000 4003 BC80   4558 5420 454E 5452   0009 FFFF FFFF 0440   0001 0000 0000 0003       @   EXT ENTR         @
   10048B00  4D41 494E 2E20 2020   2020 2020 2020 2020   244D 4149 4E2E 2020   0001 0000 0000 0080   MAIN.              $MAIN.
   10048C00  0014 0000 0000 0000   0016 0000 0002 3800   2020 5041 5620 2020   0004 0000 4000 0000                     8   PAV        @
   10048D00  0065 0000 4001 0000   0000 FFFF FFFF FFFF   0000 0000 4001 0000   0000 0000 4001 0000       @                 @          @
   10048E00  0000 0000 0002 3800   000C 1F1C 000C 1F1C   000C 1F1C 000C 1F1C   000C 1F1C 000C 1F1C         6
```

Figure E-1. Dump of a Controllee File

MODULE HEADER TABLE

The module header table contains general information concerning the object module and provides a linkage to all the other tables in the module:

Word		
1	ΔMODULEΔ 64	
2	Length 16	0 48
3	Module Name 64	
4	Date + Time Created 64	
5	T Length 16	Processor 48
6	C Length 16	Data Base Length 48
7	Type 16	Pointer 48
8	Type 16	Pointer 48

Word 3	Name of module in ASCII, expressed as 8 characters, left justified and blank filled
Word 4	Date and time module was created, in packed decimal with a positive sign. Date and time format is: year, year, month, month, day, day, hour, hour, minute, minute, second, second, millisecond, millisecond, millisecond
Word 5	Word length of tables, excluding code, followed by ASCII name of processor that created module
Word 6	Word length of code, followed by bit length of data base area. The maximum size of the data base is one large page.
Word 7 & on	Each word contains a table type and an address pointer to a table of that type. The pointer contains a bit address relative to the first word address of the header. By convention, the first table described is the code, and the second is the external/entry table. If HEX type is 0004, the pointer contains the bit address of the next module header table. Each table type is described in detail in this section.

Type	Module Name	Description
0001	CODE	Code Block Table
0002	EXT ENTR	External/Entry Table
0003	REL CODE	Code Relocation Table
0005	XFER SYM	Transfer Symbol Table
0006	SYMB TAB	Debug Symbol Table
0101	INT DATA	Interpretive Data Initialization Table
0201	INT RELO	Interpretive Relocation Initialization Table
0301	PAV	Pseudo Address Vector Table

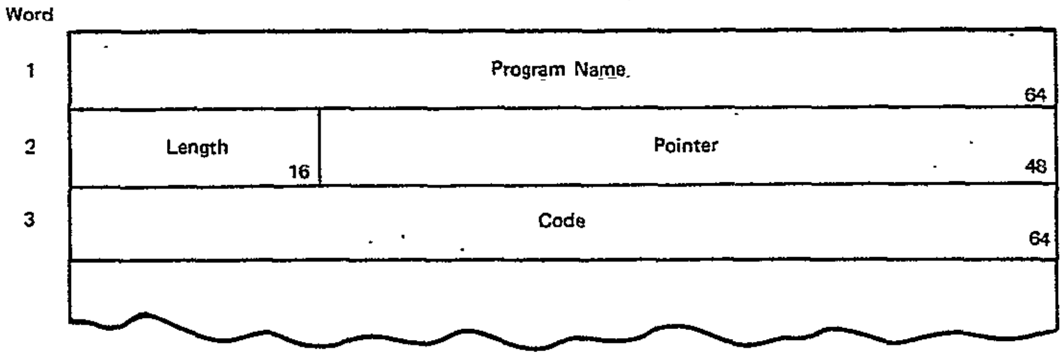
Only types 1, 2, 6, and 301 appear in the error processing information area of an object module.

CODE BLOCK TABLE

The code block table contains the executable code in the following format:

Word		
1	ΔΔCODEΔΔ 64	
2	Length 16	Back Pointer 48
3	Code	

The code block table has a pointer in the error processing information area. In this capacity the table has the following format:



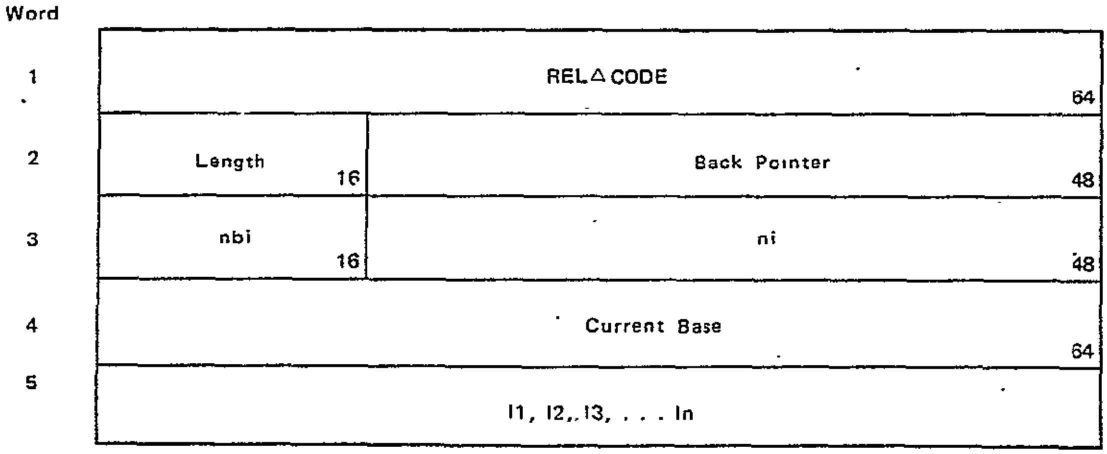
Word 1 Program name in ASCII.

Word 2 A pointer to the beginning of the error processing information area for that program.

Word 3 The executable code.
& on

CODE RELOCATION TABLE

This table describes relocation in the code itself.



Word 3 nbi is number of bits per index in the bit string starting in word 5
ni is number of indexes in the string

Word 4 Current base: current bit address to which this module is relocated

Word 5 Bit string of indexes, each nbi bits long. Each index references a half word of code to be relocated relative to the base address of the code

When this table is processed, the bit base address of the code is added to the 48-bit fields pointed to by the indexes in the bit string.

EXTERNAL/ENTRY TABLE

The external/entry table contains definitions for all entry points, external symbols, and common blocks.

Word		
1	EXT△ENTR 64	
2	Length 16	Back Pointer 48
3	m 16	n 48
4	Entry Name 1 64	
5	Entry Name 2 64	
	Entry Name m 64	
	External Name 1 64	
	External Name 2 64	
	External Name (n-m) 64	
	Entry Descriptor 1 64	
	Entry Descriptor 2 64	
	Entry Descriptor m 64	
	External Descriptor 1 64	
	External Descriptor 2 64	
	External Descriptor (n-m) 64	

Word 3	m is number of entry point names in table n is number of names in table
Word 4 through 3+m	List of entry point names
Word 4+m through 3+n	List of external names
Word 4+n through 3+m+n	List of entry point descriptors
Word 4+m+n through 3+n+n	List of external descriptors

Each descriptor is of the following form:

Type	Value
16	48

Type Field	Symbol Type	Value Field
1	Entry point in code	Relative bit address in the code
2	Entry point in data	Relative bit address in the data section
3	Constant entry point	48-bit constant
14	External procedure	0
15	External datum	0
16	Common block	Bit length of the common block

ENTRY POINTS

An entry point is a named value defined in the procedure; it is to be referenced as an external by an external procedure. It may be an address in the code block, an address in the data base, or a constant value.

COMMON BLOCKS

A common block is a named alterable space referenced by one or more procedures. A common block can be initialized with relocatable data. Blank common is a common block with name of eight blanks.

EXTERNAL PROCEDURE

An external procedure reference is used in a call. Having a symbol doubly defined as a common block and external procedure is specifically allowed. All names are eight characters, left justified and blank filled.

EXTERNAL DATA

An external datum is an external that is referenced by a method other than a procedure call.

INTERPRETIVE DATA INITIALIZATION TABLE

When the loader processes information in this table, areas of static space are initialized.

Word		
1	INT△DATA 64	
2	Length 16	Back Pointer 48
3	Data Item Descriptor Data Item 64	
.	Data Item Descriptor Data Item 64	
.		
n	Data Item Descriptor Data Item 64	

Word 3 & on	Data item descriptor and item pairs, formatted as follows:

ord1	ord2	Type	Mode	Chain
16	16	8	8	16

ord1	Pseudo address vector ordinal of static space to be initialized
ord2	Pseudo address vector ordinal relative to which relocation is to be done (relocation base)
Type	Type of data item that follows

Mode	00 Values to destination 01 Values plus relocation base to destination 02 Destination plus relocation base to destination When mode = 00, the values in the item are stored directly into the destination fields, and ord2 is ignored. When mode = 01, the relocation base is added to the values before they are stored in the destination fields. Halfword values are not defined for this case. When mode = 02, the relocation base is added to the destination fields. The value fields are absent in the various items in this case.
Chain	Relative full-word count to next data item descriptor in table

Data items may be stored in one of the following formats, depending on the type:

Data Items

Item Format 1

<table>
<tr><td>Length 16</td><td>Relative Address 48</td></tr>
<tr><td colspan="2">Value 64</td></tr>
</table>

Item Format 2

<table>
<tr><td>Length 16</td><td>Relative Address 48</td></tr>
<tr><td colspan="2">Value 64</td></tr>
<tr><td>Length2 16</td><td>Bit String 48</td></tr>
</table>

Item Format 3

<table>
<tr><td>Length 16</td><td>Relative Address 48</td></tr>
<tr><td colspan="2">Value 64</td></tr>
<tr><td>nbi 16</td><td>ni 48</td></tr>
<tr><td colspan="2">Bit String 64</td></tr>
</table>

The data item format corresponding to each type is as follows:

Type	Description	Data Item Format
1	Full-Word Broadcast	1
2	Half-Word Broadcast	1
3	Full-Word Vector Transmit	1
4	Half-Word Vector Transmit	1
5	Full-Word Sparse Vector	2
6	Half-Word Sparse Vector	2
7	Full-Word Index List	3
8	Half-Word Index List	3
9	Byte String	1
A	Bit String	1
D	Nested List	Any

For each data item type, the appropriate format is applied as follows:

FULL WORD BROADCAST

Data Item Type	1
Item Format	1
Length	Full word vector length
Value	A full word to be stored in consecutive full words starting at the relative address in the section of static space

HALF WORD BROADCAST

Data Item Type	2
Item Format	1
Length	Half-word vector length
Value	A left justified half-word to be stored in consecutive half-word locations starting at the relative bit address

FULL WORD VECTOR TRANSMIT

Data Item Type	3
Item Format	1
Length	Full-word vector length
Value	Full-word vector to be transmitted to the relative address in control section

HALF WORD VECTOR TRANSMIT

Data Item Type	4
Item Format	1
Length	Half-word vector length
Value	Half-word vector to be transmitted to the relative address in control section

FULL WORD SPARSE VECTOR

Data Item Type	5
Item Format	2
Length	Number of values in item
Value	Full-word values
Length2	Length of control vector
Bit String	Control vector having a length specified by length2

HALF WORD SPARSE VECTOR

Data Item Type	6
Item Format	2
Length	Number of values in item
Value	Left justified half word vector
Length2	Length of control vector
Bit String	Left justified control vector

FULL WORD INDEX LIST

Data Item Type	7
Item Format	3
Length	Number of values in item
Value	Full word values
nbi	Number of bits per index
ni	Number of indexes
Bit String	A bit string of ni indexes; each index is nbi bits long and contains a full-word count

HALF WORD INDEX LIST

Data Item Type	8
Item Format	3
Length	Number of values in item
Value	A left justified half-word vector
nbi	Number of bits per index
ni	Number of indexes
Bit String	A bit string of indexes; each index is nbi bits long and contains a half-word count

BYTE STRING

Data Item Type	9
Item Format	1
Length	Number of bytes in value field
Value	A left justified byte string

BIT STRING

Data Item Type	A
Item Format	1
Length	Number of bits in value field
Value	A left justified bit string

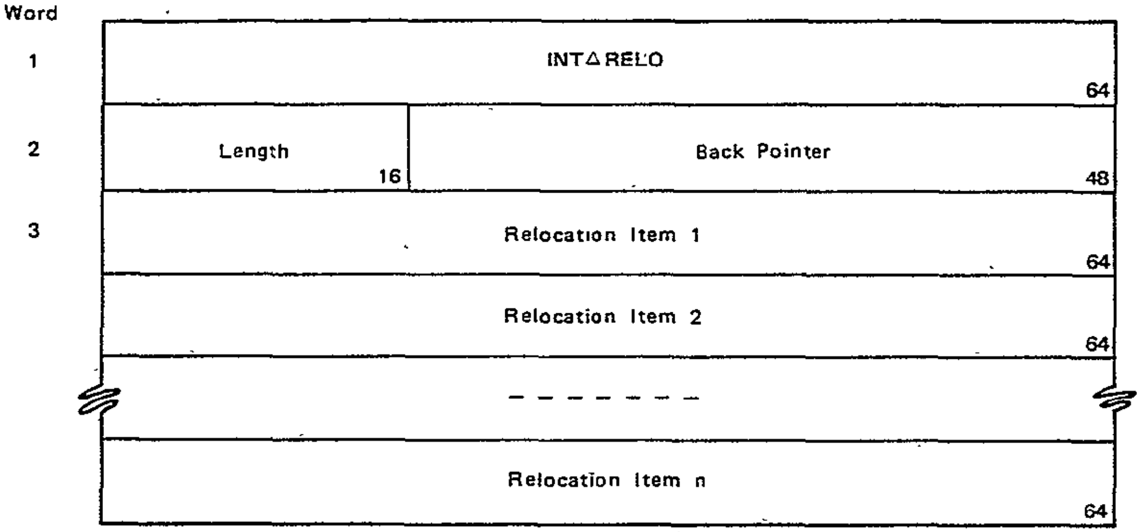
NESTED LIST

<table>
<tr><td>Ord1
16</td><td>Ord2
16</td><td>Type1
8</td><td>Mode
8</td><td>Chain1
16</td></tr>
<tr><td>Length1
16</td><td colspan="4">Rba
48</td></tr>
<tr><td>Ni2
16</td><td colspan="4">Niter
48</td></tr>
<tr><td>Ni1
16</td><td>16</td><td>Type2
8</td><td colspan="2">Chain2
24</td></tr>
<tr><td>Length2
16</td><td colspan="4">unused
48</td></tr>
<tr><td colspan="5">Value
64</td></tr>
<tr><td>Ni3
16</td><td colspan="4">Chain3
48</td></tr>
</table>

Ord1	Pseudo address vector ordinal relative to the data area to be initialized
Ord2	Pseudo address vector ordinal relative to which relocation is to be done (relocation base)
Type1	D-nested list
Mode	00 Value to destination 01 Value plus relocation base to destination 02 Destination plus relocation base to destination
Length1	Number of nested item types that follow
Rba	Relative bit address
Ni1	Nested data item
Ni2	Nested iteration start item
Ni3	Nested iteration end item
Niter	Number of times data item/items associated with this iteration start item are to be repeated
Type2	Any initialization type. If more than one data item in an iteration, types may not be mixed

Chain1	Relative full word count to next data item in nested list
Chain2	Length of data item in number of words
Chain3	0 No nested item types follow 1 More nested item types follow
Length2	Half word vector length
Value	A left justified half word to be stored in consecutive half word locations starting at the relative bit address Rba

INTERPRETIVE RELOCATION INITIALIZATION TABLE



Word 3 & on	Relocation items; item formats are similar to data initialization table formats but do not contain values

TRANSFER SYMBOL TABLE

Word		
1	XFER△SYM 64	
2	Length 16	Back Pointer 48
3	Transfer Symbol 64	

Word 3 The symbol name of the entry point to which control is to be transferred at the start of execution. The name is left justified with blank fill.

DEBUG SYMBOL TABLE

The debug symbol table, which contains the ASCII representation of symbols that appear in a program, allows a symbol to be referenced by name rather than by address. This table appears in the error processing information area if the compiler/assembler used is capable of generating the table and the appropriate option is selected and used during compilation/assembly.

Word		
1	SYMB△TAB 64	
2	Length 16	Back Pointer 48
3	Number of Symbols 16	0 48
4	Symbol 1 64	
	Symbol 2 64	
	⋮	
	Symbol n 64	

Word 2 Length of table including the symbol definition table. Back Pointer is the bit difference between word 1 of this table and word 1 of the module header table.

Word 3 Number of symbols in this table.

Word 4 & on — Symbols, which can be any of the following:

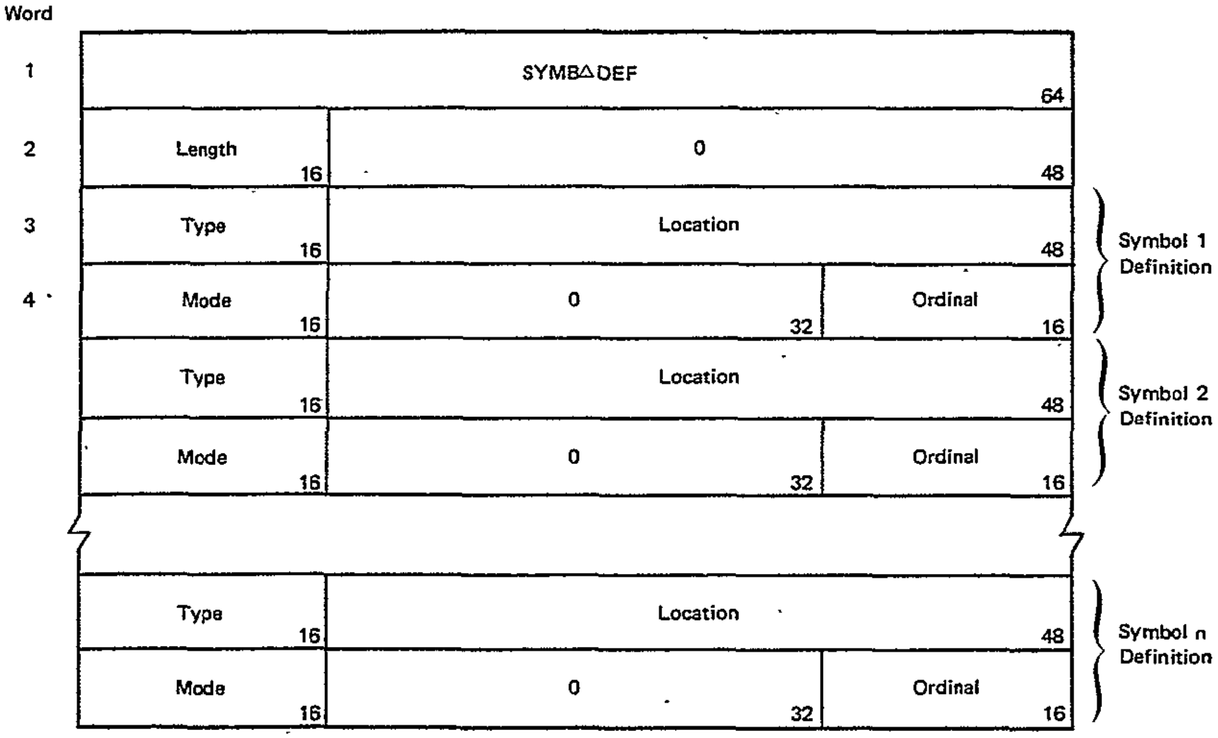
Variable or array names in ASCII; must be left-justified with blank fill.

Statement line numbers in ASCII; must be hexadecimal values, right-justified with ASCII zero fill.

Statement labels in ASCII. Labels that are symbolic names are stored left-justified with blank fill; labels that are statement numbers are stored right-justified with ASCII zero fill.

SYMBOL DEFINITION TABLE

The symbol definition table is an extension to the debug symbol table. It provides further definition to the debug symbols including the type of symbol, address, and mode.



Word 3 — Symbol type:

0 = Unknown

1 = Half-word register variable name

2 = Full-word register variable name

3 = Variable or array name

4 = Line number

5 = Label

Location field for symbol type:

1 = Half-word address within register file; since half-word values may be stored in full-word registers, location can range up to hexadecimal 1FF

2 = Full-word register number

3 = Bit address relative to the start of the data base

4 = Bit address relative to the start of the code base

5 = Bit address relative to the start of the code base

Word 4 Mode. Symbol mode, consisting of three parts: precision, description, and data type. In the case of a descriptor, P and Dtype describe the contents of the reference vector.

P	Desc	Dtype
1	3	12

P Precision base indicator:

0 = Precision base is 32-bit, or irrelevant

1 = Precision base is 64-bit

Desc Descriptor indicator:

0 = Not a descriptor

1 = Vector descriptor

2 = Vector descriptor array

4 = Sparse vector descriptor

5 = Sparse vector descriptor array

Dtype Type of the referenced vector:

0 = Unknown

1 = Logical

2 = Integer

3 = Real

4 = Complex

5 = Double precision

6 = Character

7 = Bit

Ordinal: The pseudo address vector table of the data base or common block

PSEUDO ADDRESS VECTOR TABLE
(Ordinal Description)

The table has the following format:

<table>
<tr><th>Word</th><th colspan="2"></th></tr>
<tr><td>0</td><td colspan="2">ΔΔPAVΔΔΔ 64</td></tr>
<tr><td>1</td><td>Length 16</td><td>Back Pointer 48</td></tr>
<tr><td>2</td><td colspan="2">Code Address 64</td></tr>
<tr><td>3</td><td colspan="2">Data Base Address 64</td></tr>
<tr><td>4/5</td><td colspan="2">External Address 1 64</td></tr>
<tr><td>6/7</td><td colspan="2">External Address 2 64</td></tr>
<tr><td>8/9</td><td colspan="2">External Address 3 64</td></tr>
<tr><td></td><td colspan="2">64</td></tr>
<tr><td>2n+1/2n+2</td><td colspan="2">External Address n 64</td></tr>
</table>

For common:

0 16	Address 48
0 16	Bit Length 48

For external symbol, referencing entry point in code:

0 16	Entry Address in code 48
Data Base Length 16	Data Base 48

For external symbol, referencing entry point in data:

0 16	Entry in Data Base 48
Data Base Length 16	Data Base 48

For external symbol, referencing constant entry point:

0 16	Constant Entry Value 48
Data Base Length 16	Data Base 48

APPENDIX F

SIMULATOR INPUTS AND RESULTS

```
H 32768,16,3,16                 FILTRX-AMATRX-BTRI ROUTINE                      000100
M 0,1,5,1,100,600                821   MAP RJ: GATHER RECORD                    000110
M 0,2,5,1,100,800               822   MAP XKL: GATHER RECORD                    000120
M 0,3,5,1,100,800               823   MAP YKL: GATHER RECORD                    000130
M 0,4,5,1,100,800               824   MAP ZKL: GATHER RECORD                    000140
M 0,5,5,1,98,600                922   MAP Q1': GATHER RECORD                    000160
M 0,6,5,1,98,600                923   MAP Q2': GATHER RECORD                    000170
M 0,7,5,1,98,600                924   MAP Q3': GATHER RECORD                    000180
M 0,8,5,1,100,600                925   MAP Q4': GATHER RECORD                   000190
M 0,0,5,1,98,600                926   MAP Q5': GATHER RECORD                    000200
M 0,0,5,1,98,600                932   MAP RJ': GATHER RECORD                    000210
M 0,0,5,1,100,800               942   MAP XLK': GATHER RECORD                   000220
M 0,0,5,1,100,800               952   MAP YKL': GATHER RECORD                   000230
M 0,0,5,1,16,128               962  MAP ZKL': GATHER RECORD                     000240
V 2,0,2,1,60000,12,3,1          972   XK                                        000250
V 3,0,2,1,60000,12,3,1          982   YK                                        000260
V 4,0,2,1,60000,12,3,1          992   ZK                                        000270
V 0,0,2,1,60000,12,3,1          1002  XL                                        000280
V 0,0,2,1,60000,12,3,1          1012  YL                                        000290
V 0,0,2,1,60000,12,3,1          1022  ZL                                        000300
V 0,0,3,1,60000,9,4,1           1032  T1                                        000310
V 0,0,3,1,60000,9,4,1           1042  T2                                        000320
V 0,0,3,1,60000,9,4,1           1052  T1                                        000330
V 0,0,2,1,60000,9,4,2           1053  XX(1) AND XX(2)                           000340
V 0,0,3,1,60000,9,4,1           1063  T2                                        000350
V 0,0,2,1,60000,9,4,2          1065  XX(3) AND XX(4)                            000360
V 0,0,2,1,60000,9,2,2           1082  D(1,2) AND D(1,3)                         000370
V 0,0,2,1,60000,9,2,2           1102  D(1,1) AND D(1,4)                         000380
V 5,0,1,1,60000,18,2,2          1142  RR=1/Q1                                   000390
V 0,0,0,1,60000,20,3,1          1142  DIV 2ND PASS                              000400
V 0,0,2,1,60000,9,3,2           1162  U AND V                                   000410
V 0,0,3,1,60000,9,4,1           1182  U*D12 + V*D13 = T1                        000420
V 0,0,3,1,60000,15,4,2          1183  T1 + RR*QR*D14                            000430
V 0,0,3,1,60000,9,4,1           1192  T1                                        000440
V 0,0,2,1,60000,9,3,1           1193  UT                                        000450
V 0,0,2,1,60000,15,3,1          1202  C1                                        000460
V 0,0,2,1,60000,15,3,1          1212  C2                                        000470
V 0,0,2,1,60000,9,4,2           1232  C3 AND C4                                 000480
V 0,0,2,1,60000,9,3,2           1252  C5 AND C6                                 000490
V 0,0,1,1,60000,9,2,1           1282  C7                                        000500
M 0,0,2,1,1,60000               1283  D(1,5)                                    000510
V 0,0,3,1,60000,9,4,1           1292  D(2,1)                                    000520
V 0,0,3,1,60000,15,4,1          1302  D(2,2)                                    000530
V 0,0,3,1,60000,9,4,1           1312  D(2,3)                                    000540
V 0,0,3,1,60000,9,4,1           1322  D(2,4)                                    000550
V 0,0,3,1,60000,9,4,1           1342  D(3,1)                                    000560
V 0,0,3,1,60000,9,4,1           1352  D(3,2)                                    000570
V 0,0,3,1,60000,15,4,1          1362  D(3,3)                                    000580
V 0,0,3,1,60000,9,4,1           1372  D(3,4)                                    000590
V 0,0,2,1,60000,9,4,2           1382  D(2,5) AND D(3,5)                         000600
V 0,0,3,1,60000,9,4,1           1392  D(4,1)                                    000610
V 0,0,3,1,60000,9,4,1           1402  D(4,2)                                    000620
V 0,0,3,1,60000,9,4,1           1412  D(4,3)                                    000630
V 0,0,3,1,60000,15,4,1          1422  D(4,4)                                    000640
V 0,0,1,1,60000,9,2,1           1432  D(4,5)                                    000650
V 0,0,3,1,60000,15,4,1          1442  D(5,1)                                    000660
V 0,0,3,1,60000,9,4,1           1452  D(5,2)                                    000670
V 0,0,3,1,60000,9,4,1           1462  D(5,3)                                    000680
V 0,0,3,1,60000,9,4,1           1472  D(5,4)                                    000690
V 0,0,2,1,60000,9,3,1           1482  END AMATRX                                000700
V 0,0,1,1,60000,20,2,1          1522  RMJ=RM/RJ                                 000710
V 0,0,0,1,60000,18,3,1          1522  DIV 2ND PASS                              000720
V 0,0,2,1,60000,8,3,2           1542  RR AND RF                                 000730
```

```
M 0,0,5,1,98,588                   1652  MAP F1(N):GATHER                     000740
R 5                              1550  DO 23                                  000750
R 5                              1560  DO 22                                  000760
V 0,0,0,1,60000,9,2,2             1602  A(N,M) AND B(N,M)                     000770
M 0,0,2,1,1,60000                 1612  MAP B(N,M): SCATTER                   000780
C                                 END  DO 22                                  000790
M 0,0,2,1,1,60000                 1632  MAP B(N,M): SCATTER                   000800
V 0,0,2,1,60000,9,4,2             1642  A(N,M) AND C(N,M)                     000810
C                                 END  DO 23                                  000820
F 32768,16,3,16                 BTRI ROUTINE LINES 4330 TO 6100               000840
V 0,0,1,1,600,18,2,2          4330  L11=1/B(1,1)                              000850
V 0,0,0,1,600,18,2,1                DIVIDE 2ND PASS                           000860
M 0,1,1,1,1,600               4340  MAP: L21=B1(2,1)                          000870
V 1,0,3,1,600,9,4,2           4350 & 4360  U12=B1(1,2)*L11: T1=B1-L21*U12     000880
V 0,0,1,1,600,18,2,2          4360  L22=1./T1                                 000890
V 0,0,0,1,600,18,2,1                DIVIDE 2ND PASS                           000900
V 0,0,2,1,600,9,3,2           4380  U13, U14                                  000910
V 0,0,1,1,600,9,3,1           4390  U15, L31                                  000920
V 0,0,1,1,600,9,2,1           4410  L32                                       000930
V 0,0,3,1,600,12,4,1          4420 U23                                        000940
V 0,0,3,1,600,9,4,1           4430  TEMP1                                     000950
V 0,0,1,1,600,9,2,1           4430  TEMP2                                     000960
V 0,0,1,1,600,18,2,2          4430  DIVIDE 1ST PASS                           000970
V 0,0,0,1,600,18,2,1          4430  L33                                       000980
V 0,0,3,1,600,9,4,1           4440  U24                                       000990
V 0,0,3,1,600,9,4,1           4450  U25                                       001000
M 0,2,1,1,1,600               4460  L41                                       001010
V 2,0,3,1,600,12,4,2          4470  L42, T1                                   001020
V 0,0,3,1,600,15,4,1          4480 L43                                        001030
V 0,0,3,1,600,9,4,1           4490 T1=L41*U13+L32*U24                         001040
V 0,0,2,1,600,12,3,1                U34=(B1(3,4)-T1)*L33                      001050
V 0,0,3,1,600,9,4,1           4500  T1=U14*L41+U24*L42                        001060
V 0,0,3,1,600,9,4,1           4500  T1=B1(4,4)-T1-(U34*L43)                   001070
V 0,0,1,1,600,18,2,2          4500  L44=1./T1                                 001080
V 0,0,0,1,600,18,2,1                DIVIDE 2ND PASS                           001090
V 0,0,3,1,600,9,4,1           4512 T1                                         001100
V 0,0,2,1,600,12,2,1          4513  U35                                       001110
M 0,2,1,1,1,600               4522 L51                                        001120
V 2,0,3,1,600,12,4,2          4542  L52 AND T1=L51*U13                        001130
V 0,0,3,1,600,12,4,2          4544 L53                                        001140
V 0,0,3,1,600,9,4,2           4552 T1                                         001150
V 0,0,3,1,600,9,4,2           4553 L54                                        001160
V 0,0,3,1,600,12,4,1          4562  T1=L42*U25-L43*U35                        001170
V 0,0,3,1,600,12,4,1          4563 T1                                         001180
V 0,0,3,1,600,12,4,2          4572 U45 AND T1                                 001190
V 0,0,3,1,600,12,4,1          4574 T2                                         001200
V 0,0,3,1,600,12,4,1          4575 T1                                         001210
V 0,0,1,1,600,18,2,2          4576  L51=1./T1                                 001220
V 0,0,0,1,600,18,2,1           DIVIDE 2ND PASS                                001230
V 0,0,3,1,600,9,4,2           4602 & 4603  D1 AND T1                          001240
V 0,0,3,1,600,9,4,2           4612  D2 AND T1                                 001250
V 0,0,3,1,600,12,4,1          4614 D3                                         001260
V 0,0,3,1,600,12,4,1          4622 T1                                         001270
V 0,0,3,1,600,12,4,1          4623 T1                                         001280
V 0,0,3,1,600,9,4,2           4632 D4                                         001290
V 0,0,3,1,600,12,4,2          4634 T2                                         001300
V 0,0,3,1,600,12,4,2          4635 T1                                         001310
V 0,0,0,1,600,9,3,2           4652 D5 AND F15                                 001320
V 0,0,3,1,600,12,4,2          4672 F14 AND T1                                 001330
V 0,0,3,1,600,12,4,1          4674 F13                                        001340
V 0,0,3,1,600,12,4,1          4682 T1                                         001350
V 0,0,3,1,600,12,4,1          4683 F12                                        001360
V 0,0,3,1,600,12,4,1          4692 T1                                         001370
```

```
V 0,0,3,1,600,12,4,1        4693 T2
V 0,0,3,1,600,12,4,1        4694 F11
R 5
V 0,0,3,1,600,12,4,2         4732   D1
V 0,0,3,1,600,12,4,2        4742 D2 AND T1
V 0,0,3,1,600,12,4,1        4744 D3
V 0,0,3,1,600,15,4,1        4762 T1
V 0,0,3,1,600,15,4,1        4762 T1
V 0,0,3,1,600,12,4,2        4764 D4 AND T1
V 0,0,3,1,600,15,4,1        4766 T2
V 0,0,3,1,600,15,4,1        4767 T1
V 0,0,1,1,600,9,3,2         4772  D5 AND D15
V 0,0,3,1,600,12,4,2        4792 B4M AND T1
V 0,0,3,1,600,15,4,1        4794 B3M
V 0,0,3,1,600,15,4,1        4802 T1
V 0,0,3,1,600,15,4,1        4803 B2M
V 0,0,3,1,600,15,4,1        4812 T1
V 0,0,3,1,600,15,4,1        4813 T2
V 0,0,2,1,600,12,3,1        4814 B1M
C                            END DO 12
R 1                          DO IMAX TIMES
R 5                          DO 14 LOOP
V 0,0,3,1,600,15,4,1        4922 T1
V 0,0,3,1,600,15,4,1        4923 T2
V 0,0,3,1,600,15,4,1        4924 T1
V 0,0,1,1,600,9,2,1         4925 F1
C                            END DO LOOP 14
R 25                         DO LOOP 11
V 0,0,3,1,600,15,4,1        4972 T1
V 0,0,3,1,600,15,4,1        4973 T2
V 0,0,3,1,600,15,4,1        4974 T1
V 0,0,1,1,600,9,2,1         4975 HNM
C                            END DO LOOP 11
V 0,0,1,1,600,18,2,2        4330  L11=1/B(1,1)
V 0,0,0,1,600,18,2,1              DIVIDE 2ND PASS
M 0,1,1,1,1,600             4340  MAP; L21=B1(2,1)
V 1,0,3,1,600,9,4,2         4350 & 4360  U12=B1(1,2)*L11; T1=B1-L21*U12
V 0,0,1,1,600,18,2,2        4360  L22=1./T1
V 0,0,0,1,600,18,2,1              DIVIDE 2ND PASS
V 0,0,2,1,600,9,3,2         4380  U13, U14
V 0,0,1,1,600,9,3,1         4390  U15, L31
V 0,0,1,1,600,9,2,1         4410  L32
V 0,0,3,1,600,12,4,1        4420 U23
V 0,0,3,1,600,9,4,1         4430  TEMP1
V 0,0,1,1,600,9,2,1         4430  TEMP2
V 0,0,1,1,600,18,2,2        4430  DIVIDE 1ST PASS
V 0,0,0,1,600,18,2,1        4430  L33
V 0,0,3,1,600,9,4,1         4440  U24
V 0,0,3,1,600,9,4,1         4450  U25
M 0,2,1,1,1,600             4460  L41
V 2,0,3,1,600,12,4,2        4470  L42, T1
V 0,0,3,1,600,15,4,1        4480 L43
V 0,0,3,1,600,9,4,1         4490 T1=L41*U13+L32*U24
V 0,0,2,1,600,12,3,1              U34=(B1(3,4)-T1)*L33
V 0,0,3,1,600,9,4,1         4500  T1=U14*L41+U24*L42
V 0,0,3,1,600,9,4,1         4500  T1=B1(4,4)-T1-(U34*L43)
V 0,0,1,1,600,18,2,2        4500  L44=1./T1
V 0,0,0,1,600,18,2,1              DIVIDE 2ND PASS
V 0,0,3,1,600,9,4,1         4512 T1
V 0,0,2,1,600,12,2,1        4513  U35
M 0,2,1,1,1,600             4522 L51
V 2,0,3,1,600,12,4,2        4542  L52 AND T1=L51*U13
V 0,0,3,1,600,12,4,2        4544 L53
```

```
V 0,0,3,1,600,9,4,2          4552 T1
V 0,0,3,1,600,9,4,2          4553 L54
V 0,0,3,1,600,12,4,1         4562  T1=L42*U25-L43*U35
V 0,0,3,1,600,12,4,1         4563 T1
V 0,0,3,1,600,12,4,2         4572 U45 AND T1
V 0,0,3,1,600,12,4,1         4574 T2
V 0,0,3,1,600,12,4,1         4575 T1
V 0,0,1,1,600,18,2,2         4576  L51=1./T1
V 0,0,0,1,600,18,2,1          DIVIDE 2ND PASS
V 0,0,3,1,600,9,4,2          4602 & 4603  D1 AND T1
V 0,0,3,1,600,9,4,2          4612  D2 AND T1
V 0,0,3,1,600,12,4,1         4614 D3
V 0,0,3,1,600,12,4,1         4622 T1
V 0,0,3,1,600,12,4,1         4623 T1
V 0,0,3,1,600,9,4,2          4632 D4
V 0,0,3,1,600,12,4,2         4634 T2
V 0,0,3,1,600,12,4,2         4635 T1
V 0,0,0,1,600,9,3,2          4652 D5 AND F15
V 0,0,3,1,600,12,4,2         4672 F14 AND T1
V 0,0,3,1,600,12,4,1         4674 F13
V 0,0,3,1,600,12,4,1         4682 T1
V 0,0,3,1,600,12,4,1         4683 F12
V 0,0,3,1,600,12,4,1         4692 T1
V 0,0,3,1,600,12,4,1         4693 T2
V 0,0,3,1,600,12,4,1         4694 F11
R 5
V 0,0,3,1,600,12,4,2          4732   D1
V 0,0,3,1,600,12,4,2         4742 D2 AND T1
V 0,0,3,1,600,12,4,1         4744 D3
V 0,0,3,1,600,15,4,1         4762 T1
V 0,0,3,1,600,15,4,1         4762 T1
V 0,0,3,1,600,12,4,2         4764 D4 AND T1
V 0,0,3,1,600,15,4,1         4766 T2
V 0,0,3,1,600,15,4,1         4767 T1
V 0,0,1,1,600,9,3,2          4772  D5 AND D15
V 0,0,3,1,600,12,4,2         4792 B4M AND T1
V 0,0,3,1,600,15,4,1         4794 B3M
V 0,0,3,1,600,15,4,1         4802 T1
V 0,0,3,1,600,15,4,1         4803 B2M
V 0,0,3,1,600,15,4,1         4812 T1
V 0,0,3,1,600,15,4,1         4813 T2
V 0,0,2,1,600,12,3,1         4814 B1M
C                             END DO 12
C          END  IMAX LOOP
R 5                           DO 14 LOOP
V 0,0,3,1,600,15,4,1         4922 T1
V 0,0,3,1,600,15,4,1         4923 T2
V 0,0,3,1,600,15,4,1         4924 T1
V 0,0,1,1,600,9,2,1          4925 F1
C                             END DO LOOP 14
R 25                          DO LOOP 11
V 0,0,3,1,600,15,4,1         4972 T1
V 0,0,3,1,600,15,4,1         4973 T2
V 0,0,3,1,600,15,4,1         4974 T1
V 0,0,1,1,600,9,2,1          4975 HNM
C                             END DO LOOP 11
V 0,0,1,1,600,18,2,2         4330  L11=1/B(1,1)
V 0,0,0,1,600,18,2,1               DIVIDE 2ND PASS
M 0,1,1,1,1,600              4340  MAP; L21=B1(2,1)
V 1,0,3,1,600,9,4,2          4350 & 4360  U12=B1(1,2)*L11; T1=B1-L21*U12
V 0,0,1,1,600,18,2,2         4360  L22=1./T1
V 0,0,0,1,600,18,2,1               DIVIDE 2ND PASS
V 0,0,2,1,600,9,3,2          4380  U13, U14
V 0,0,1,1,600,9,3,1          4390  U15, L31
V 0,0,1,1,600,9,2,1          4410  L32
V 0,0,3,1,600,12,4,1         4420 U23
V 0,0,3,1,600,9,4,1          4430  TEMP1
V 0,0,1,1,600,9,2,1          4430  TEMP2
V 0,0,1,1,600,18,2,2         4430  DIVIDE 1ST PASS
V 0,0,0,1,600,18,2,1         4430  L33
V 0,0,3,1,600,9,4,1          4440  U24
V 0,0,3,1,600,9,4,1          4450  U25
M 0,2,1,1,1,600              4460  L41
V 2,0,3,1,600,12,4,2         4470  L42, T1
```

```
V 0,0,3,1,600,15,4,1       4480 L43                                                   002750
V 0,0,3,1,600,9,4,1        4490 T1=L41*U13+L32*U24                                    002760
V 0,0,2,1,600,12,3,1            U34=(B1(3,4)-T1)*L33                                  002770
V 0,0,3,1,600,9,4,1        4500  T1=U14*L41+U24*L42                                   002780
V 0,0,3,1,600,9,4,1        4500  T1=B1(4,4)-T1-(U34*L43)                              002790
V 0,0,1,1,600,18,2,2       4500  L44=1./T1   :                                        002800
V 0,0,0,1,600,18,2,1             DIVIDE 2ND PASS                                      002810
V 0,0,3,1,600,9,4,1        4512 T1                                                    002820
V 0,0,2,1,600,12,2,1       4513  U35                                                  002830
M 0,2,1,1,1,600            4522 L51                                                   002840
V 2,0,3,1,600,12,4,2       4542  L52 AND T1=L51*U13                                   002850
V 0,0,3,1,600,12,4,2       4544 L53                                                   002860
V 0,0,3,1,600,9,4,2        4552 T1                                                    002870
V 0,0,3,1,600,9,4,2        4553 L54                                                   002880
V 0,0,3,1,600,12,4,1       4562  T1=L42*U25-L43*U35                                   002890
V 0,0,3,1,600,12,4,1       4563 T1                                                    002900
V 0,0,3,1,600,12,4,2       4572 U45 AND T1                                            002910
V 0,0,3,1,600,12,4,1       4574 T2                                                    002920
V 0,0,3,1,600,12,4,1       4575 T1                                                    002930
V 0,0,1,1,600,18,2,2       4576  L51=1./T1                                            002940
V 0,0,0,1,600,18,2,1        DIVIDE 2ND PASS                                           002950
V 0,0,3,1,600,9,4,2        4602 & 4603  D1 AND T1                                     002960
V 0,0,3,1,600,9,4,2        4612  D2 AND T1                                            002970
V 0,0,3,1,600,12,4,1       4614 D3                                                    002980
V 0,0,3,1,600,12,4,1       4622 T1                                                    002990
V 0,0,3,1,600,12,4,1       4623 T1                                                    003000
V 0,0,3,1,600,9,4,2        4632 D4                                                    003010
V 0,0,3,1,600,12,4,2       4634 T2                                                    003020
V 0,0,3,1,600,12,4,2       4635 T1                                                    003030
V 0,0,0,1,600,9,3,2        4652 D5 AND F15                                            003040
V 0,0,3,1,600,12,4,2       4672 F14 AND T1                                            003050
V 0,0,3,1,600,12,4,1       4674 F13                                                   003060
V 0,0,3,1,600,12,4,1       4682 T1                                                    003070
V 0,0,3,1,600,12,4,1       4683 F12                                                   003080
V 0,0,3,1,600,12,4,1       4692 T1                                                    003090
V 0,0,3,1,600,12,4,1       4693 T2                                                    003100
V 0,0,3,1,600,12,4,1       4694 F11                                                   003110
R 1                      DO IMAX TIMES                                                003120
R 5                         DO 14 LOOP                                                003130
V 0,0,3,1,600,15,4,1       4922 T1                                                    003140
V 0,0,3,1,600,15,4,1       4923 T2                                                    003150
V 0,0,3,1,600,15,4,1       4924 T1                                                    003160
V 0,0,1,1,600,9,2,1        4925 F1                                                    003170
C                           END DO LOOP 14  ______                                    003180
C                  END IMAX LOOP                                                      003190
E                                                                                     003200


                 ****************************************
                 *                                      *
                 *      FLOATING POINT SAVEVALUES       *
                 *                                      *
                 ****************************************

          NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
          VECFP   6.9739465968500E-001     VECBZ    8.6175078107600E-001
          CLKPD                 728981


          NUMBER ............. CONTENTS    NUMBER ............. CONTENTS:
          MAPFP   1.1522933080280E-004     MAPBZ    7.6949735576900E-001
```

```
H 32768,16,3,16                                                                                   000100
F                                   THIS IS RIGHT HAND SIDE                                       000110
M 0,0,5,1,1,60000                           330  RJ                                               000120
M 0,7,5,1,1,60000                           340  XKJ                                              000130
M 0,8,5,1,1,60000                           350  YKJ                                              000140
M 0,9,5,1,1,60000                           360  ZKJ                                              000150
M 0,0,5,1,1,60000                           370  Q1                                               000160
M 0,0,5,1,1,60000                           380  Q2                                               000170
M 0,0,5,1,1,60000                           390  Q3                                               000180
M 0,0,5,1,1,60000                           400  Q4                                               000190
M 0,0,5,1,1,60000                           410  Q5                                               000200
V 7,1,2,1,60000,12,3,1                      420  XK                                               000210
V 8,2,2,1,60000,12,3,1                      430  YK                                               000220
V 9,3,2,1,60000,12,3,1                      440  ZK                                               000230
V 0,4,2,1,60000,12,3,1                      450  XL                                               000240
V 0,5,2,1,60000,12,3,1                      460  YL                                               000250
V 0,6,2,1,60000,12,3,1                      470  ZL                                               000260
V 00,20,3,1,60000,15,4,1                    480   XX(1)                                           000270
V 0,21,3,1,60000,15,4,1                     490    TEMP2                                          000280
V 0,22,2,1,60000,9,3,2                      490    XX(1) , XX(2)                                  000290
V 0,23,3,1,60000,15,4,1                     500   TEMP1                                           000300
V 22,24,3,1,60000,15,4,1                    510   TEMP2                                           000310
V 0,0,2,1,60000,9,3,2                       510   XX(3), XX(4)                                    000320
V 0,0,1,1,60000,18,2,2                      550  RR=1./Q                                          000330
V 0,0,0,1,60000,18,2,1                      550  64 BIT DIVIDE                                    000340
V 0,0,2,1,60000,9,4,2                       560 AND 570 U AND V                                   000350
V 0,0,3,1,60000,12,4,2                      590 W                                                 000360
V 0,0,2,1,60000,15,4,1                      590  T                                                000370
V 0,0,3,1,60000,12,4,2                      590 FINISH QS,FORM F1(1) AND T1                       000380
V 0,0,3,1,60000,15,4,1                      600 U**2+V**2                                         000390
V 0,0,3,1,60000,15,4,1                      600 W**2+T1*Q1                                        000400
V 0,0,3,1,60000,15,4,1                      600 P=GAMI*(Q5-.5*T1)                                 000410
V 0,0,3,1,60000,15,4,1                      620 F1(2)                                             000420
V 0,0,3,1,60000,15,4,1                      630 F1(3)                                             000430
V 0,0,3,1,60000,15,4,1                      640 F1(4)                                             000440
V 0,0,1,1,60000,9,2,1                       650 (Q5+PP)                                           000450
V 0,0,3,1,60000,15,4,1                      650 F1(5)                                             000460
R S                                        DO 20 LOOP                                             000470
V 0,0,2,1,60000,12,3,1                      710 S2                                                000480
C                                                                                                 000485
V 7,1,2,1,60000,12,3,1                      420  XK                                               000490
V 8,2,2,1,60000,12,3,1                      430  YK                                               000500
V 9,3,2,1,60000,12,3,1                      440  ZK                                               000510
V 0,4,2,1,60000,12,3,1                      450  XL                                               000520
V 0,5,2,1,60000,12,3,1                      460  YL                                               000530
V 0,6,2,1,60000,12,3,1                      470  ZL                                               000540
V 00,20,3,1,60000,15,4,1                    480   XX(1)                                           000550
V 0,21,3,1,60000,15,4,1                     490    TEMP2                                          000560
V 0,22,2,1,60000,9,3,2                      490    XX(1) , XX(2)                                  000570
V 0,23,3,1,60000,15,4,1                     500   TEMP1                                           000580
V 22,24,3,1,60000,15,4,1                    510   TEMP2                                           000590
V 0,0,2,1,60000,9,3,2                       510   XX(3), XX(4)                                    000600
V 0,0,1,1,60000,18,2,2                      550  RR=1./Q                                          000610
V 0,0,0,1,60000,18,2,1                      550  64 BIT DIVIDE                                    000620
V 0,0,2,1,60000,9,4,2                       560 AND 570 U AND V                                   000630
V 0,0,3,1,60000,12,4,2                      590 W                                                 000640
V 0,0,2,1,60000,15,4,1                      590  T                                                000650
V 0,0,3,1,60000,12,4,2                      590 FINISH QS,FORM F1(1) AND T1                       000660
V 0,0,3,1,60000,15,4,1                      600 U**2+V**2                                         000670
V 0,0,3,1,60000,15,4,1                      600 W**2+T1*Q1                                        000680
V 0,0,3,1,60000,15,4,1                      600 P=GAMI*(Q5-.5*T1)                                 000690
V 0,0,3,1,60000,15,4,1                      620 F1(2)                                             000700
V 0,0,3,1,60000,15,4,1                      630 F1(3)                                             000710
```

```
V 0,0,3,1,60000,15,4,1                   640 F1(4)                                              000720
V 0,0,1,1,60000,9,2,1                    650 (Q5+PP)                                            000730
V 0,0,3,1,60000,15,4,1                   650 F1(5)                                              000740
R 5                                     DO 20 LOOP                                              000750
V 0,0,2,1,60000,12,3,1                   710 S2                                                 000760
C                                                                                               000765
V 7,1,2,1,60000,12,3,1                   420  XK                                                000770
V 8,2,2,1,60000,12,3,1                   430  YK                                                000780
V 9,3,2,1,60000,12,3,1                   440  ZK                                                000790
V 0,4,2,1,60000,12,3,1                   450  XL                                                000800
V 0,5,2,1,60000,12,3,1                   460  YL                                                000810
V 0,6,2,1,60000,12,3,1                   470  ZL                                                000820
V 00,20,3,1,60000,15,4,1                 480   XX(1)                                            000830
V 0,21,3,1,60000,15,4,1                  490    TEMP2                                           000840
V 0,22,2,1,60000,9,3,2                   490    XX(1) , XX(2)                                   000850
V 0,23,3,1,60000,15,4,1                  500   TEMP1                                            000860
V 22,24,3,1,60000,15,4,1                 510   TEMP2                                            000870
V 0,0,2,1,60000,9,3,2                    510   XX(3), XX(4)                                     000880
V 0,0,1,1,60000,18,2,2                   550  RR=1./Q                                           000890
V 0,0,0,1,60000,18,2,1                   550  64 BIT DIVIDE                                     000900
V 0,0,2,1,60000,9,4,2                    560 AND 570 U AND V                                    000910
V 0,0,3,1,60000,12,4,2                   590 W                                                  000920
V 0,0,2,1,60000,15,4,1                   590  T                                                 000930
V 0,0,3,1,60000,12,4,2                   590 FINISH QS,FORM F1(I) AND T1                        000940
V 0,0,3,1,60000,15,4,1                   600 U**2+V**2                                          000950
V 0,0,3,1,60000,15,4,1                   600 W**2+T1*Q1                                         000960
V 0,0,3,1,60000,15,4,1                   600 P=GAMI*(Q5-.5*T1)                                  000970
V 0,0,3,1,60000,15,4,1                   620 F1(2)                                              000980
V 0,0,3,1,60000,15,4,1                   630 F1(3)                                              000990
V 0,0,3,1,60000,15,4,1                   640 F1(4)                                              001000
V 0,0,1,1,60000,9,2,1                    650 (Q5+PP)                                            001010
V 0,0,3,1,60000,15,4,1                   650 F1(5)                                              001020
R 5                                     DO 20 LOOP                                              001030
V 0,0,2,1,60000,12,3,1                   710 S2                                                 001040
C                                                                                               001045
E                                                                                               001050


                     *****************************************
                     *                                       *
                     *       FLOATING POINT SAVEVALUES       *
                     *                                       *
                     *****************************************


          NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
          VECFP    1.0333437718820E+000     VECBZ    9.0235653319400E-001
          CLKPD                  772976


          NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
          MAPFP    7.2770688160800E-007     MAPBZ    2.6197447737840E-001
```

```
H 32768,16,3,16                                                                        000001
F                                                                                      000040
F                                                                                      000041
F                                                                                      000042
F                    SIMULATION INPUT FOR VISMAT SUBROUTINE                            000043
F                                                                                      000044
F                                                                                      000045
M 0,0,1,1,1,60000           2070  D(2,5)=0.0                                           000050
M 0,0,1,1,1,60000           2080  DU(2,5)=0.0                                          000051
M 0,0,1,1,1,60000           2190  D(3,5)=0.0                                           000052
M 0,0,1,1,1,60000           2200  DU(3,5)=0.0                                          000053
M 0,0,1,1,1,60000           2310  D(4,5)=0.0                                           000054
M 0,0,1,1,1,60000           2320  DU(4,5)=0.0                                          000055
F                                                                                      000100
F                                                                                      000120
V 0,0,3,1,60000,9,2,2       1740  T0=(ZZ1*ZZ1)+(ZZ2*ZZ2); T3=(ZZ2*ZZ2)                 000130
V 0,0,2,1,60000,9,2,2       1740  T2=ZZ1*ZZ1; T4=ZZ3*ZZ3                               000140
V 0,0,2,1,60000,9,4,2       1750  T1=(R3*VNU)*RJ; 1780  T5=ZZ1*ZZ2                     000150
V 0,0,2,1,60000,12,3,1      1740  S0=(T0+T4)*RJ                                        000160
V 0,0,3,1,60000,12,4,2      1750  S1=(S0+(T1*T2)); 1810  S0=S0*(GKPR*GKAP)             000170
V 0,0,2,1,60000,9,3,2       1790  T6=ZZ1*ZZ3; 1800  T7=ZZ2*ZZ3                         000180
V 0,0,2,1,60000,9,3,2       1760  T13=T1*T3; 1780  S4=T1*T5                            000190
V 0,0,3,1,60000,9,4,2       1770  S3=(S0+(T1*T4)); 1760  S2=S0+T13                     000200
V 0,0,2,1,60000,9,3,2       1790  S5=T1*T6; 1800  S6=T1*T7                             000210
V 0,0,1,1,60000,9,2,1       1820  E=QS*RR                                              000220
V 0,0,2,1,60000,9,4,2       1840  C0=S0(*,*,1:LMAX-1)+S0(*,*,2:LMAX); 1850  C1=  000230
V 0,0,2,1,60000,9,4,2       1860  C2=S2(*,*,1:LMAX-1)+S2(*,*,2:LMAX); 1870  C3=  000240
V 0,0,2,1,60000,9,4,2       1880  C4=S4(*,*,1:LMAX-1)+S4(*,*,2:LMAX); 1890  C5=  000250
F                                                                                      000260
F                    DEFINE STATEMENTS GO HERE                                         000270
F                                                                                      000280
V 0,0,3,1,60000,9,4,1       1970  T1=((C1*U(*,*,1:LMAX-1))+(C4*V(*,*,1:LMAX-1))) 000290
V 0,0,2,1,60000,9,4,2       1970  T3=T1+C5*W(*,*,1:LMAX-1); 207^  D(2,5)= 0.0       000300
V 0,0,2,1,60000,9,4,2       1980  D(2,1)=-T3*RR(*,*,1:LMAX-1); 1900  C6=            000310
V 0,0,3,1,60000,9,4,1       1990  T1=((C1*U(*,*,1:LMAX-1))+(C4*V(*,*,1:LMAX-1))   000320
V 0,0,3,1,60000,15,4,1      2000  DU(2,1)=RR*(-T1+(-C5*W(*,*,2:LMAX)))               000330
V 0,0,2,1,60000,9,3,2       2010  D(2,2)=C1*RR(,,1:LMAX-1); 2020  DU(2,2)=C1*RR(,000340
V 0,1,2,1,60000,9,3,2       2030  D(2,3)=C4*RR; 2040  DU(2,3)=C4*RR(,,2:LMAX)         000350
M 1,0,1,1,1,60000           2130  D(3,2)=D(2,3)                                        000360
M 0,0,1,1,1,60000           2140  DU(3,2)=DU(2,3)                                      000370
V 0,2,2,1,60000,9,3,2       2050  D(2,4)=C5*RR(,,1:LMAX-1); 2060  DU(2,4)=C5*RR    000380
M 2,0,1,1,1,60000           2250  D(4,2)=D(2,4)                                        000381
M 0,0,1,1,1,60000           2260  DU(4,2)=DU(2,4)                                      000382
V 0,0,3,1,60000,9,4,1       2090  T1=(C2*V(*,*,1:LMAX-1))+(C4*U(*,*,1:LMAX-1))    000410
V 0,0,3,1,60000,15,4,1      2100  D(3,1)=RR*(-T1+(-C6*W(*,*,1:LMAX-1)))             000420
V 0,0,3,1,60000,9,4,1       2110  T1=(C2*V(*,*,2:LMAX))+(C4*U(*,*,2:LMAX))          000430
V 0,0,3,1,60000,15,4,1      2120  DU(3,1)=RR*(-T1+(-C6*W(*,*,2:LMAX)))               000440
V 0,0,2,1,60000,9,3,2       2150  D(3,3)=C2*RR(*,*,1:LMAX-1); 2160 DU(3,3)=C2*RR 000450
V 0,3,2,1,60000,9,3,2       2170  D(3,4)=C6*RR; 2180  DU(3,4)=C6*RR(*,*,2:LMAX)    000460
M 3,0,1,1,1,60000           2270  D(4,3)=D(3,4)                                        000461
M 0,0,1,1,1,60000           2280  DU(4,3)=DU(3,4)                                      000462
V 0,0,3,1,60000,9,4,1       2210  T1=(C3*W(*,*,1:LMAX-1))+(C5*W(*,*,1:LMAX-1))    000490
V 0,0,3,1,60000,15,4,1      2220  D(4,1)=RR*(-T1+(-C6*V(*,*,1:LMAX-1)))             000500
V 0,0,3,1,60000,9,4,1       2230  T1=(C3*W(*,*,2:LMAX))+(C5*W(*,*,1:LMAX-1))       000510
V 0,0,3,1,60000,15,4,1      2240  DU(4,1)=RR*(-T1+(-C6*V(*,*,2:LMAX)))               000520
V 0,0,2,1,60000,9,3,2       2290  D(4,4)=C3*RR; 2300  DU(4,4)=C3*RR(,,2:LMAX)         000530
F                                                                                      000540
F                                                                                      000550
F                                                                                      000560
V 0,0,2,1,60000,12,3,2      2330  T1=(C1-C2);   T2=T1*U(*,*,1:LMAX-1)                  000570
V 0,0,2,1,60000,12,3,2      2340  T3=(C2-C0);   T4=T3*V(*,*,1:LMAX-1)                  000580
V 0,0,2,1,60000,12,3,2      2340  T5=(C3-C0);   T6=T5*W(*,*,1:LMAX-1)                  000590
V 0,0,2,1,60000,9,3,2       2350  T7=C4*V;   T8=2*U(*,*,1:LMAX-1)                      000600
```

```
V 0,0,2,1,60000,9,4,1       2330  T9=T2*(-U(*,*,1:LMAX-1))+T4*(-V(*,*,1:LMAX-1)) 000610
V 0,0,2,1,60000,9,4,2       2350  T10=2*U;   T11=C5*W(*,*,1:LMAX-1)               000620
V 0,0,3,1,60000,9,4,1       2350  T12=(T7*(-T10))+(T8*(-T11))                      000630
V 0,0,2,1,60000,9,4,2       2360  T13=C6*V(*,*,1:LMAX-1);    T14=2*W               000640
V 0,0,3,1,60000,9,4,1       2360-70  T15=(T13*(-T14))+(C0*E(*,*,1:LMAX-1))        000650
V 0,0,3,1,60000,15,3,1      2340  T16=T9+(T6*W(*,*,1:LMAX-1))+T12                  000660
V 0,0,3,1,60000,15,4,1      2370  D(5,1)=((T16+T11)+T15)*RR(*,*,1:LMAX-1)          000670
F                                                                                 000680
F                                                                                 000690
V 0,0,2,1,60000,9,2,2       2380  T16=T1*U(*,*,2:LMAX); T15=T16*U(*,*,2:LMAX)     000700
V 0,0,2,1,60000,9,2,2       2380  T14=T2*V(*,*,2:LMAX); T1=T14*V(*,*,2:LMAX)      000710
V 0,0,2,1,60000,9,2,2       2380  T3=T5*W(*,*,2:LMAX);  T5=T3*W                    000720
V 0,0,2,1,60000,9,3,2       2390  T10=C4*V(*,*,2:LMAX);  T8=2*U(*,*,2:LMAX)       000730
V 0,0,2,1,60000,9,4,2       2390  T9=2*U(*,*,2:LMAX);  T12=C5*W(*,*,2:LMAX)       000740
V 0,0,3,1,60000,9,4,1       2390  T17=(T10*(-T9))+(T8*(-T12))                      000750
V 0,0,2,1,60000,9,4,2       2390  T18=C6*V(*,*,2:LMAX);  T19=2*W(*,*,2:LMAX)      000760
V 0,0,3,1,60000,9,4,1       2390  T20=(T18*T(-T19))+(C0*E(*,*,2:LMAX))            000770
V 0,0,2,1,60000,12,4,1      2390  T19=T20+T17+T15                                  000780
V 0,0,3,1,60000,15,3,1      2290  DU(5,1)=((T1*T5)+T19)*RR(*,*,2:LMAX)            000790
F                                                                                 000800
F                                                                                 000810
V 0,0,3,1,60000,15,4,1      2410  D(5,2)=(T2+T7+T11)*RR(*,*,1:LMAX-1)             000820
V 0,0,3,1,60000,15,4,1      2430  DU(5,2)=(T15+T10+T12)*RR(*,*,2:LMAX)            000830
V 0,0,3,1,60000,9,4,1       2440  T7=(C4*U(*,*,1:LMAX-1))+(C6*W(*,*,1:LMAX-1))    000840
V 0,0,2,1,60000,12,3,1      2450  D(5,3)=(T4+T7)*RR(*,*,1:LMAX-1)                 000850
V 0,0,3,1,60000,9,4,1       2460  T7=(C4*U(*,*,2:LMAX))+(C6*W(*,*,2:LMAX))        000860
V 0,0,2,1,60000,12,3,1      2470  DU(5,3)=(T1+T7)*RR(*,*,2:LMAX)                  000870
V 0,0,3,1,60000,9,4,1       2480  T7=(C5*U(*,*,1:LMAX-1))+(C6*V)                  000880
V 0,0,2,1,60000,12,3,1      2490  D(5,4)=(T6+T7)*RR(*,*,1:LMAX-1)                 000890
V 0,0,3,1,60000,9,4,1       2500  T7=(C5*U(*,*,2:LMAX))+(C6*V)                    000900
V 0,0,2,1,60000,12,3,1      2510  DU(5,4)=(T5+T7)*RR(*,*,2:LMAX)                  000910
V 0,0,2,1,60000,9,3,2       D(5,5)=C0*RR(,,1:LMAX-1); 2520  DU(5,5)=C0*RR(,,2:LMA000920
F                                                                                 000930
F                                                                                 000940
F         DO  LOOP  N=2,5  M=1,5                                                  000950
F                                                                                 000960
F                                                                                 000970
R 4                                                                               000980
R 5                                                                               000990
F                                                                                 001000
F                   DEFINE STATEMENTS GO HERE                                     001010
F                                                                                 001020
V 0,0,2,1,60000,9,3,1       2570  A(N,M)=A(N,M)+DRE*D1(*,*,2:LMAX)                001030
V 0,0,3,1,60000,12,4,1      2580 B(N,M)=B(N,M)-DRE*(D1(,,3:LMAX)+DU1(,,2:LMAX)    001040
V 0,0,2,1,60000,9,3,1       2590  C(N,M)=C(N,M)+DRE*DU1(*,*,3:LMAX)               001050
C                                                                                 001060
C                                                                                 001070
E                                                                                 001080

                        **************************************
                        *                                    *
                        *      FLOATING POINT SAVEVALUES     *
                        *                                    *
                        **************************************


               NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
               VECFP   1.1846927223380E+000      VECBZ    9.9933263653100E-001
               CLKPD                 930621



               NUMBER ............. CONTENTS     NUMBER ............. CONTENTS:
               MAPFP   8.0591341655600E-007      MAPBZ    9.6709609986800E-002
```

```
H 32768,16,3,16          MUTUR ROUTINE                                          000100
V 0,0,3,1,60000,9,2,1           1170  TZZ=ZZ1*ZZ1+ZZ2*ZZ2                       000110
V 0,0,3,1,60000,9,2,1           1170  SCIS=ABS(TZZ+ZZ3*ZZ3)                     000120
M 0,0,1,1,1,60000              1110  SNOR(*,1,*)=0.0                            000130
V 0,0,1,1,60000,20,2,2          1190  SQ1=B/SQ1                                 000140
V 0,0,0,1,60000,18,3,1          DIVIDE SECOND PASS                              000150
V 0,0,1,1,60000,9,2,1           Y(1)=A+SQ1                                      000160
M 0,0,1,1,1,60000              S1(*,*,*)=0.0                                    000161
M 0,0,1,1,1,60000              S2(*,*,*)=0.0                                    000162
M 0,0,1,1,1,60000              S3(*,*,*)=0.0                                    000163
R 4                           MAXIMUM OF 4 ITERATIONS                           000170
V 0,0,1,1,60000,20,2,2          X/Y(I)                                          000180
V 0,0,0,1,60000,18,3,1          DIVIDE SECOND PASS                              000190
V 0,0,2,1,60000,12,3,1          Y(I+1)=0.5*(Y(I)+(X/Y(I)))                      000200
C                                                                               000210
V 0,0,1,1,60000,20,2,2          SCAL=1./SQRT(SCIS)                              000220
V 0,0,0,1,60000,18,3,1          DIVIDE SECOND PASS                              000230
R 16                          L=2,LMAX-1                                        000240
M 0,0,2,1,6,100                MSNOR(*,*)=SNOR(*,L,*)                           000250
V 0,0,1,1,600,9,2,2           1210  NSNOR(*,*)=SNOR(*,L,*)=MSNOR(*,*)+SCAL 000260
V 0,0,2,1,600,12,3,1          1220  SNORA(*,L,*)=0.5*(NSNOR(*,*)+MSNOR(*,*)000270
C                                                                               000280
V 0,0,3,1,60000,9,4,1        440  T1A=Q4(*,2:LMAX+1,*)*RL1-Q4(*,*,*)*RL         000290
V 0,0,3,1,60000,9,4,1        450  T1B=Q3(*,2:LMAX+1,*)*RL1-Q3(*,*,*)*RL         000300
V 0,0,2,1,60000,9,4,2        440/450  T1C/D=ZZ2/3(*,2:LMAX+1,*)+ZZ2/3(*,*,      000310
M 0,0,2,1,6,100                1140  MAA1(*,*)=ZZ1(*,1,*)                       000320
M 0,0,2,1,6,100                1140  MAA2(*,*)=ZZ2(*,1,*)                       000330
M 0,0,2,1,6,100                1140  MAA3(*,*)=ZZ3(*,1,*)                       000340
V 0,0,3,1,60000,9,4,1        440  T1E=(T1C*T1A)-(T1D*T1B)                       000360
V 0,0,3,1,60000,9,4,1        470  T2A=Q2*RL1-Q2*RL                              000370
V 0,0,2,1,60000,9,4,1        440  T1=0.5*T1E : 480  T2C=ZZ1(*,*,*)+ZZ1(*,2      000380
V 0,0,3,1,60000,9,4,1        470  T2E=(T1D*T2A)-(T2C*T1A)                       000390
V 0,0,3,1,60000,9,4,1        500  T3E=(T2C*T1B)-(T1C*T2A)                       000400
V 0,0,2,1,60000,9,4,1        470/500  T2=0.5*T2E:  T3=0.5*T3E                   000410
F                                                                               000420
F                        REDEFINE  RL & RL1                                     000430
F                                                                               000440
V 0,0,3,1,60000,9,4,1         590  TA1=Q4(*,*,3:JSL+2)*RL1-Q4(*,*,*)*RL         000450
V 0,0,3,1,60000,9,4,1         600  TA2=Q3(*,*,3:JSL+2)*RL1-Q3(*,*,*)*RL         000460
V 0,0,3,1,60000,9,4,1         590  TA=XX(2)*TA1-XX(3)*TA2                       000470
V 0,0,3,1,60000,9,4,1         610  TB1=Q2(*,*,3:JSL+2)*RL1-Q2(*,*,*)*RL         000480
V 0,0,3,1,60000,9,4,1         610  TB=XX(3)*TB1+XX(1)*TA1                       000490
V 0,0,3,1,60000,9,4,1         630  TC=XX(1)*TA2-XX(2)*TB1                       000500
F                                                                               000510
F                        REDEFINE  RL & RL1                                     000520
F                                                                               000530
V 0,0,3,1,60000,9,4,1         660  TD1=Q4(3:KMAX+2,*,*)*RL1-Q4(*,*,*)*RL        000540
V 0,0,3,1,60000,9,4,1         670  TD2=Q3(3:KMAX+2,*,*)*RL1-Q3(*,*,*)*RL        000550
V 0,0,3,1,60000,9,4,1         660  TD=YY(2)*TD1-YY(3)*TD2                       000560
V 0,0,3,1,60000,9,4,1         680  TE1=Q2(3:KMAX+2,*,*)*RL1-Q2(*,*,*)*RL        000570
V 0,0,3,1,60000,9,4,1         680  TE=YY(3)*TE1-YY(1)*TD1                       000580
V 0,0,3,1,60000,9,4,1         700  TF=YY(1)*TD2-YY(2)*TE1                       000590
V 0,0,3,1,60000,15,4,1        720  S1=S1+0.25*(TA+TD)                           000600
V 0,0,3,1,60000,15,4,1        730  S2=S2+0.25*(TB+TE)                           000610
V 0,0,3,1,60000,15,4,1        740  S3=S3+0.25*(TC+TF)                           000620
F                                                                               000630
F                        REDEFINE  RL1 & RL                                     000640
F                                                                               000650
V 0,0,3,1,60000,9,4,1         780  TA1=Q4(*,3:LM2,3:JSL+2)*RL1-Q4(*,3:LM2,*     000660
V 0,0,3,1,60000,9,4,1         790  TA2=Q3(*,3:LM2,3:JSL+2)*RL1-Q3(*,3:LM2,*     000670
V 0,0,3,1,60000,9,4,1         780  TA=XX(2)*TA1-XX(3)*TA2                       000680
V 0,0,3,1,60000,9,4,1         800  TB1=Q2(*,3:LM2,3:JSL+2)*RL1-Q2(*,3:LM2,*     000690
V 0,0,3,1,60000,9,4,1         800  TB=XX(3)*TB1+XX(1)*TA1                       000700
V 0,0,3,1,60000,9,4,1         820  TC=XX(1)*TA2-XX(2)*TB1                       000710
M 0,0,2,1,6,100                  1050  MQ6(*,*)=Q6(*,1,*)                       000720
M 0,0,2,1,6,100                  1050  MQ1(*,*)=Q1(*,1,*)                       000730
M 0,0,1,1,1,60000             1120  KM2(*,*,*)=1                                000740
M 0,0,1,1,1,60000             1130  YDUM(*,*,*)=1.E-3                           000750
```

```
F                                                                       000760
F                             REDEFINE  RL & RL1                        000770
F                                                                       000780
V 0,0,3,1,60000,9,4,1           860  TD1=Q4(3:KM2,3:LM2,*)*RL1-Q4(*,3:LM2,*)* 000790
V 0,0,3,1,60000,9,4,1           870  TD2=Q3(3:KM2,3:LM2,*)*RL1-Q3(*,3:LM2,*)* 000800
V 0,0,3,1,60000,9,4,1           860  TD=YY(2)*TD1-YY(3)*TD2               000810
V 0,0,3,1,60000,9,4,1           880  TE1=Q2(3:KM2,3:LM2,*)*RL1-Q2(*,3:LM2,*)* 000820
V 0,0,3,1,60000,9,4,1           880  TE=YY(3)*TE1-YY(1)*TD1               000830
V 0,0,3,1,60000,9,4,1           900  TF=YY(1)*TD2-YY(2)*TE1               000840
V 0,0,3,1,60000,15,4,1          920  S1=S1+0.25*(TA+TD)                   000880
V 0,0,3,1,60000,15,4,1          930  S2=S2+0.25*(TB+TE)                   000890
V 0,0,3,1,60000,15,4,1          940  S3=S3+0.25*(TC+TF)                   000900
V 0,0,2,1,60000,9,4,2           950  TW1=T1+S1 !  TW2=T2+S2               000910
V 0,0,3,1,60000,9,2,1           950  TW1=TW1*TW1+TW2*TW2                  000920
V 0,0,3,1,60000,15,2,1          950  TW=TW1+(T3+T3)*(T3+T3)               000930
V 0,0,3,1,60000,9,2,1           970  U1=Q2*Q2+Q3*Q3                       000940
V 0,0,2,1,60000,9,2,1           970  U1=U1+Q4*Q4                          000950
M 0,0,2,1,6,100                        1300  MYDU(*,*)=YDU(*,1,*)         000980
F                             SQRT OF TW & UTOT                         001010
F                             I.E.  SQRT(X)                             001020
V 0,0,2,1,60000,9,3,2           SQ1=C+X                                 001030
V 0,0,1,1,60000,20,2,2          SQ1=B/(C+X)                             001040
V 0,0,0,1,60000,18,3,1          DIVIDE SECOND PASS                      001050
V 0,0,1,1,60000,20,2,2          SQ1=B/(C+X)  FOR UTOT                   001060
V 0,0,0,1,60000,18,3,1          DIVIDE SECOND PASS                      001070
V 0,0,2,1,60000,9,3,2           Y(1)=A+SQ1                              001080
M 0,0,1,1,1,60000               990  TURMU(*,*,*)=0.0                   001090
R 2                           FOR TW & UTOT                             001100
R 4                           MAXIMUM OF 4 ITERATIONS FOR NECESSARY ACCURAC  001110
V 0,0,1,1,60000,20,2,2          X/Y(I)                                  001120
V 0,0,0,1,60000,18,3,1          SECOND PASS                             001130
V 0,0,2,1,60000,12,3,1          Y(I+1)=0.5*(Y(I)+X/Y(I))                001140
C                                                                       001150
M 0,0,2,1,6,100                        1040  MTAU(*,*)=TAS(*,1,*)         001160
M 0,0,1,1,60000,1               1410  MTAS(*,*,*)=TAS(KM2(*,*,*)-1)     001170
M 0,0,1,1,60000,1               1420  MTAS(*,*,*)=TAS(KM2(*,*,*)+1)     001180
C                             TAS=ABS(SQRT(TW))                         001190
V 0,0,1,1,60000,20,2,2          UU=SQRT(UTOT)/Q1                        001200
M 0,0,2,1,6,100                        1270  MUU(*,*)=UU(*,1,*)           001210
V 0,0,2,1,600,15,3,1              1050  TRA1=MQ6*MQ1*MTAU               001230
V 0,0,3,1,600,9,2,1               1140  TSCIS=MZZ1*MZZ1+MZZ2*MZZ2       001240
V 0,0,3,1,600,12,3,1              1140 TYM=C+TYM+MZZ3*MZZ3              001250
V 0,0,3,1,600,12,4,1              1050  TRA1=C+TRA1*RE*(1/26.)          001260
V 0,0,2,1,600,9,3,1               1050/1140  Y(1)=A+TRA1 ! Y(1)=A+TYM   001270
R 2                              SQRT FOR RA AND YM                     001280
R 4                              MAXIMUM OF 4 ITERATIONS.               001290
V 0,0,1,1,600,20,2,2              X/Y(I)                                001300
V 0,0,0,1,600,18,2,1              SECOND PASS DIVIDE                    001310
V 0,0,2,1,600,12,3,1              Y(I+1)=0.5*(Y(I)+X/Y(I))              001320
C                                                                       001330
C                                1050 RA                                001340
V 0,0,1,1,600,20,2,2              1140  YM=0.5/SQRT(TYM)                001350
V 0,0,0,1,600,18,3,1              SECOND PASS FOR DIVIDE                001360
V 0,0,1,1,60000,9,2,1                1250  EX1=SNORA(*,2:LMAX+1,*)-RA(*,2:LMAX+001361
R 1                              CALCULATION OF EXP(EX1) EXACTLY AS DONE ABOVE001362
V 0,0,3,1,60000,15,4,2                                                  001363
V 0,0,2,1,60000,9,3,1                                                   001364
V 0,0,3,1,60000,15,3,1                                                  001365
V 0,0,3,1,60000,15,3,1                                                  001366
V 0,0,2,1,60000,9,3,1                                                   001367
V 0,0,2,1,60000,9,3,1                                                   001368
V 0,0,1,1,60000,20,2,2                                                  001369
V 0,0,0,1,60000,18,3,1                                                  001370
C                                                                       001371
V 0,0,3,1,60000,15,4,1              1240  YDU(,2:LMAX+1,)=SNORA*TAS*(1.0-EXP(E001372
R 16                             DO 21 L=2,LM                           001373
M 0,0,1,1,6,100               MUU(*,*)=UU(*,L,*)                        001374
V 0,0,0,1,600,9,2,0              1270  IF(MUU.LT.UMAX(*,1,*))           001380
V 0,0,0,1,600,9,1,1                      THEN MUMIN(*,*)=MUU            001390
C                                                                       001395
R 5                              DO 18 L=2,LEDGE                        001400
M 0,0,1,1,6,100               MUU(*,*)=UU(*,L,*)                        001410
V 0,0,0,1,600,9,2,0              1290  IF(MUU.GT.MUMAX) ...             001420
V 0,0,0,1,600,9,1,1                       THEN  MUMAX=MUU               001430
M 0,0,1,1,6,100               MYDU(*,*)=YDU(*,L,*)                      001440
V 0,0,0,1,600,9,2,0              1300 BIT=MYDU .LT. MYDUM               001450
V 0,0,0,1,600,9,1,1                       IF (BIT) THEN  MYDUM = MYDU   001460
```

```
M 0,0,1,1,6,100          MSNORA(*,*)=SNORA(*,L,*)                        001470
V 0,0,0,1,600,9,1,1          1320  IF (BIT) THEN  MYM=MSNORA             001480
M 0,0,1,1,6,100          MKM2(*,*)=KM2(*,L,*)                            001490
V 0,15,0,1,600,9,1,1         1330  IF (BIT) THEN  MKM2=L-1               001500
C                              18 CONTINUE                               001510
M 15,0,1,1,60000,1           MSNOR0(*,*,*)=SNOR(KM2(*,*,*)+2)            001511
M 0,2,1,1,60000,1            MSNOR1(*,*,*)=SNOR(KM2(*,*,*)+1)            001512
M 0,0,1,1,60000,1            MSNOR2(*,*,*)=SNOR(KM2(*,*,*))              001513
M 0,3,1,1,60000,1            MSNOR3(*,*,*)=SNOR(KM2(*,*,*)-1)            001514
V 0,0,0,1,60000,9,4,0          TBIT1=KM2(*,*,*).LT.2; TBIT2=KM2.GT.LEDGE-001520
T 2                                                                      001530
V 2,0,2,1,60000,9,4,2         1390  TYM3=(MSNOR1+MSNOR0); TYM1=SNOR2+SNOR001540
V 3,0,2,1,60000,9,3,1         1400  YM1=0.5*TYM1  1390  YM3=0.5*TYM3     001550
V 0,0,2,1,60000,9,3,2          1410/1420  EX1=YM1-RA & EX2=YM3-RA        001560
R 2                          TWO EXPONENTIALS TO CALCULATE  EX1 & EX2    001570
V 0,0,3,1,60000,15,4,2         N=EX*(1/LOG(2))*16; K=N/16-1(OR 0)        001580
V 0,0,2,1,60000,9,3,1          (2**K)*(2**(M/16)                         001590
V 0,0,3,1,60000,15,3,1         Q=Q01*F*F+Q00                             001600
V 0,0,3,1,60000,15,3,1         P=P01*F*F+P00                             001610
V 0,0,2,1,60000,9,3,1          F1=Q+F*P                                  001620
V 0,0,2,1,60000,9,3,1          F2=Q-F*P                                  001630
V 0,0,1,1,60000,20,2,2         2**F=F1/F2                                001640
V 0,0,0,1,60000,18,3,1         DIVIDE SECOND PASS.                       001650
C                                                                        001660
R 30                         60000 BITS;  SCALAR UNIT DOES 64 AT A TIME. 001670
L                                                                        001680
L                                                                        001690
F                                                                        001700
L                            1380  BIT= TBIT1.OR.TBIT2                   001710
C                                                                        001720
V 0,0,3,1,60000,15,4,1         1410  YDUM1=YM1*MTAS*(1.0-EXP(EX1))       001730
V 0,0,3,1,60000,15,4,1         1420  YDUM3=YM3*MTAS*(1.0-EXP(EX2))       001740
V 0,0,2,1,60000,9,4,2          1430/1440  C2=YDUM-YDUM1 & C3=YDUM3-YDUM1 001750
V 0,0,2,1,60000,9,3,2          1450/1460  DY2=YM-YM1 & DY3=YM3-YM        001760
V 0,0,2,1,60000,15,2,2         1470  TAM1=DY2*DY2; TAM2=DY3*DY3          001770
V 0,0,3,1,60000,9,4,1          1470  TAM1=TAM1*C2+TAM2*C3                001780
V 0,0,3,1,60000,15,2,1         1470  TAM2=DY2*DY3*(DY2+DY3)              001790
V 0,0,3,1,60000,9,4,1          1480  TBM=DY2*C3+DY3*C2                   001800
V 0,0,1,1,60000,20,2,2         1470  AM=TAM1/TAM2                        001810
V 0,0,0,1,60000,18,3,1         DIVIDE 2ND PASS                           001820
V 0,0,1,1,60000,20,2,2         1480  BM=TBM/TAM2                         001830
V 0,0,0,1,60000,9,2,0          1490  BIT1=BM.GE.0                        001840
T 2                                                                      001850
R 30                                                                     001860
L                                                                        001870
L                                                                        001880
F                                                                        001890
L                            1500  BIT=(.NOT.BIT.OR.NOT.BM)              001900
C                                                                        001910
V 0,0,1,1,60000,20,2,2         AM/BM                                     001920
V 0,0,0,1,60000,18,3,1         DIVIDE 2ND PASS                           001930
V 0,0,2,1,60000,9,3,1          1510  IF(BIT) THEN  Y600=YM-0.5*(AM/BM)   001940
V 0,0,3,1,60000,15,4,1         1520 IF(BIT) THEN  TDU=YDUM-0.25*AM*(AM/BM001950
V 0,0,0,1,60000,9,4,0          1530  TBIT1=YDU.LT.YDUM;  TBIT2=Y600.LT.YM001960
V 0,0,0,1,30000,9,2,0          1530  TBIT3=Y600.GT.YM3 ; EACH HALF DONE I001970
T 2                                                                      001980
R 30                                                                     001990
L                                                                        002000
L                                                                        002010
L                                                                        002020
F                                                                        002030
F                                                                        002040
L                            1530  BIT1=TBIT1.OR.TBIT2.OR.TBIT3          002050
C                                                                        002060
R 30                                                                     002070
L                                                                        002080
L                                                                        002090
F                                                                        002100
L                            BIT=BIT.AND..NOT.BIT1                       002110
C                                                                        002120
V 0,0,0,1,60000,9,1,1          1550  IF(BIT) YDUM=YDU                    002130
V 0,0,0,1,60000,9,1,1          1560  IF(BIT) YM= Y600                    002140
V 0,0,0,1,60000,9,2,0          1570  YM .GT. MSNOR1 &  YM .LT. SNOR2     002150
V 0,0,1,1,60000,9,2,1          1570  IF(...) KM2=KM2+1                   002160
```

```
V 0,0,1,1,60000,9,2,1         1580  IF(...) KM2=KM2-1                   002170
V 0,0,3,1,58800,15,4,2        1620  SNOR(*,1:LEDGE,*)=.5*(SNOR(*,1:LEDGE002180
F     +SNOR(*,2*LEDGE+1,*)); 1680  TFIA=FKLEB*SNOR(*,1:LEDGE,*)        002190
V 0,0,2,1,58800,15,3,1        1630  FFC=K*Q1(,1:LEDGE,)*Q6(,1:LEDGE,)   002200
V 0,0,2,1,600,15,3,1        1640  T1=YM*YDUM & UDIFF=ABS(UMAX-UMIN)     002210
V 0,0,2,1,58800,9,3,1         1640  TWO(*,1:LEDGE,*)=FCC*T1  &  FFCWK=FC002220
V 0,0,0,1,600,9,3,1         1670  IF( YDUM .GT. UDIFF*YDUMF)...         002230
V 0,0,2,1,600,12,3,0        1670  T1=YM*UDIFF*UDIFF                     002240
V 0,0,1,1,600,20,2,2              T1/YDUM                             002250
V 0,0,0,1,600,18,3,1       DIVIDE,SECOND PASS                           002260
V 0,0,1,1,58800,9,2,1         TWO=FFCWK*(T1/YDUM)                       002270
V 0,0,1,1,58800,20,2,2        1680  T1=SNOR(,1:LED,)/YM                 002280
V 0,0,0,1,58800,18,3,1        DIVIDE 2ND PASS                           002290
V 0,0,1,1,58800,9,2,1         1680  FIA=FKLEB*T1                        002300
V 0,0,0,1,58800,9,2,1         1690  IF(FIA.GT.1E5)...                   002310
V 0,0,0,1,58800,9,2,1         1690   THEN  FIA=1.E5                     002320
V 0,0,2,1,58800,15,1,1        1700  T1=FIA*FIA*FIA                      002330
V 0,0,3,1,58800,15,3,1        1700  FI=1.0+5.5*(T1*T1)                  002340
V 0,0,1,1,58800,20,2,2        1710  TWO=TWO/FI                          002350
V 0,0,0,1,58800,18,3,1        DIVIDE 2ND PASS                           002360
F                                                                       002370
F EXP(-RA*SNOR)                                                         002380
V 0,0,1,1,60000,9,2,1         1770  EX1=-RA*SNOR                        002390
V 0,0,3,1,60000,15,4,2                                                  002400
V 0,0,2,1,60000,9,3,1                                                   002410
V 0,0,3,1,60000,15,3,1                                                  002420
V 0,0,3,1,60000,15,3,1                                                  002430
V 0,0,2,1,60000,9,3,1                                                   002440
V 0,0,2,1,60000,9,3,1                                                   002450
V 0,0,1,1,60000,20,2,2                                                  002460
V 0,0,0,1,60000,18,3,1                                                  002470
V 0,0,3,1,60000,15,4,1        1770  T1=.4*SNOR*(1-EXP(EX1))             002480
V 0,0,2,1,60000,9,3,1         1770  T2=Q6*Q1*RE                         002490
V 0,0,1,1,60000,9,2,2         1770  TM1=ABS(T1+T2);  1830  TURMU(*,1:LED002500
V 0,0,0,1,30000,9,2,0         1850  BIT=TMI.LE.TWO                      002510
V 0,0,0,1,30000,9,2,1           IF(BIT) THEN TURMU = TWO                002520
M 0,0,2,1,6,100             1910  SMP=TURMU(*,1,*)                      002530
R 17                       DO 60 L=2,LMAX                               002540
M 0,0,2,1,6,100             1930  TURMS=TURMU(*,L,*)                    002550
V 0,0,2,1,600,12,3,1        1940  TURMU(*,L,*)=2.0*SMP-TURMS            002560
M 0,0,1,1,1,600             1950   60 SMP=TURMS                         002570
C                                                                       002580
E                                                                       002590
```

```
               ****************************************
               *                                      *
               *      FLOATING POINT SAVEVALUES       *
               *                                      *
               ****************************************


     NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
     VECFP   7.5499436261100E-001     VECBZ    8.5181743077500E-001
     CLKPD                1529465


     NUMBER ............. CONTENTS    NUMBER ............. CONTENTS:
     MAPFP   1.9635051323680E-002     MAPBZ    3.5302342841140E-001
```

```
H 32768,16,3,16           LX ROUTINE                                                    000090
F                                $DTDX=DT*DX1                                           000100
R 1                              $DO 1 K=KS1,KE2                                        000110
R 1                              $DO 2 J=JS1,JE2                                        000120
R 5                                                                                     000130
M 0,0,1,1,100,1                   $PROICT(     )=                                       000140
C                                                                                       000150
V 0,0,2,1,100,15,2,1              $P=A*B*C                                              000160
R 2                              $DO 4 N=1,2                                            000170
F                                                                                       000180
F                                                                                       000190
F                                                                                       000200
V 0,0,2,1,101,12,2,1              $UII=(A+B)*C                                          000210
V 0,1,2,1,101,9,2,1               $TMP=(A*B-C)                                          000220
V 0,0,3,1,101,15,3,0              $TMP=(A*B-C)*TMP>0  -CONTROL VECT                     000230
V 0,0,1,1,100,9,2,0               $U(I+1)-U(I)<0 -CONTROL VECT                          000240
M 0,0,4,1,100,25                  $IF(      ) UII=                                      000250
M 0,0,4,1,100,25                  $IF(      ) UII=                                      000260
F                                $U(1) =                                                000270
F                                                                                       000280
V 0,0,2,1,100,9,2,2               $RLMBDA=A*B$RK=C*D                                    000300
F                                $C=.5*DY1                                              000340
V 0,0,2,1,100,12,2,1              $DYX=(A+B)*C                                          000350
V 0,0,2,1,100,9,4,2               $UYX=A-B$VYX=C-D                                      000360
V 0,0,3,1,100,12,2,1              $(W-W)*DZ                                             000370
V 0,0,3,1,100,12,2,1              $(U-U)*DX                                             000380
V 0,0,3,1,100,9,2,1               $VXY*DY+T                                             000390
V 0,0,2,1,100,9,3,1               $UXY*DY-T                                             000400
V 0,0,3,1,100,9,3,1               $LMBDA+2*MU                                           000410
V 0,0,2,1,100,12,2,1              $(V-V)*DX                                             000440
V 0,0,3,1,100,9,3,1               $T-VYX*DYX                                            000450
V 0,0,2,1,100,9,2,1               $UYX*DY+T                                             000460
V 0,0,2,1,100,12,3,1              $(W-W)*DYX                                            000480
V 0,0,3,1,100,12,2,1              $(W-W)*DX                                             000490
V 0,0,3,1,100,12,2,1              $(U-U)*DZ                                             000500
V 0,0,2,1,100,15,3,1              $T+T-T                                                000510
V 0,0,3,1,100,12,2,1              $(E-E)*DX                                             000540
V 0,0,2,1,100,12,3,1              $RK*(T+T)                                             000550
V 0,0,3,1,100,15,4,1              $S*U+TAU*W                                            000560
V 0,0,3,1,100,15,4,1              $DISX=TAU*V+T+T                                       000570
V 0,0,1,1,100,9,2,1               $F(1)=A*B                                             000580
R 4                                                                                     000590
V 0,0,2,1,100,9,3,1               $F(J)=A*B+C                                           000600
C                                                                                       000610
F                                 $IF(ISMO=0) GO TO 25                                  000620
F                                 $II=II+1                                              000630
V 0,0,3,1,98,15,3,1               $T=P-2*P+P                                            000640
V 0,0,3,1,98,15,3,1               $T+2*ABS(P)+T                                         000660
V 0,0,1,1,98,9,1,1                $C*T                                                  000670
V 0,0,0,1,98,20,2,2               $ / PASS 1                                            000680
V 0,0,1,1,98,18,3,1               $ / PASS 2                                            000690
V 0,0,1,1,98,9,1,1                $G*ABS(T)                                             000700
R 29                              $CII=SQRT(G*ABS)                                     S000710
L                                 $RANGE REDUCTION                                     S000720
R 5                                                                                    S000730
F                                 $SCALAR MANIPULATION OF EXPONENTS                    S000740
C                                                                                      S000750
L                                 $                                                    S000760
C                                                                                      S000770
R 3                               $R(2,1) = P/Q     (3.66D)                            S000780
V 0,0,2,1,98,9,1,1                $P=POLY $ Q=POLY                                     S000790
C                                                                                      S000800
V 0,0,0,1,98,20,2,2               $R=P/Q                                               S000810
```

```
V 0,0,1,1,98,18,3,1                                                      S000820
R 2                                  $TWO NEWTON ITERATIONS    (14.6D)   S000830
V 0,0,0,1,98,20,2,2                  $X/Y(N)                             S000840
V 0,0,1,1,98,18,3,1                                                      S000850
V 0,0,2,1,98,12,2,1                  $.5*(Y(N)+X/Y(N))                   S000860
C                                                                        S000870
V 0,0,1,1,98,9,2,1                   $POST NORMALIZATION - END OF SQRT   S000880
V 0,0,2,1,98,12,3,1                  $COEF=(T+T)*T                        000890
R 5                                  $DO K6=1,5                           000900
V 0,0,3,1,98,15,4,1                  $F= (A+B)*C+D                        000910
C                                    $END OF FX SUBR                      000920
R 5                                  $PRDICT =                            000930
V 0,0,3,1,98,15,3,1                  $(A+B)*C+D                           000940
V 0,0,3,1,98,15,2,1                  $A*(T+C*D)                           000950
C                                                                         000960
V 0,0,0,1,98,20,2,2                  $ / PASS 1                           000970
V 0,0,1,1,98,18,3,1                  $ / PASS 2 RHOI=1/PRDICT             000980
V 0,0,2,1,98,9,3,2                   $U=A*B $B=C*B                        000990
V 0,0,2,1,98,9,3,2                   $W=A*B $ T=C*B                       001000
V 0,0,3,1,98,15,2,1                  $U*U+V*V                             001010
V 0,0,2,1,98,9,2,1                   $W*W+T                               001020
V 0,0,2,1,98,9,2,1                   $EI=-.5*T+T                          001030
V 0,0,2,1,98,15,2,1                  $P=A*B*EI                            001040
F                                    $IF(J.NE.2) GO TO 100                001080
F                                    $IF(J.LT.JE2) GO TO 35               001090
F                                    $IF(K.NE.KE2) GO TO 50               001160
C                                    $4 CONTINUE END OF N LOOP            001190
C                                    $2 CONTINUE END OF J LOOP            001230
C                                    $1 CONTINUE END OF K LOOP            001240
E                                    $END                                 001350
```

```
                    ****************************************
                    *                                      *
                    *     FLOATING POINT SAVEVALUES        *
                    *                                      *
                    ****************************************


          NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
          VECFP    7.7348901456800E-001    VECBZ    7.0556960045800E-001
          CLKPD                    2398


          NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
          MAPFP    2.3456436185400E-002    MAPBZ    2.2851781386000E-001
```

```
H 32768,16,3,16          LYI ROUTINE                                       000100
R 1                                                                        000110
R 2                                                                        000120
F                            $VLYI - SETUP PORTION                         000130
F                            $SINGLE QUOTE = KLEN                          000140
F                            $COLON = KLEN*JLEN/2                          000150
V 0,1,0,1,5000,20,1,2           $OLD DO 8 LOOP                             000160
V 1,2,1,1,5000,18,3,1           $RHOI=                                     000170
V 2,0,3,1,5000,15,3,1                                                      000180
V 0,0,3,1,5000,15,3,1                                                      000190
V 0,0,2,1,5000,9,3,2                                                       000200
V 0,0,2,1,5000,9,2,2                                                       000210
V 0,0,3,1,5000,15,3,1                                                      000220
V 0,0,3,1,5000,15,4,1           $8 CONTINUE                                000230
R 6                          $MISC                                         000240
M 0,0,1,1,1,100                  $UI*UI ETC                                000250
V 0,0,1,1,100,9,1,1                                                        000260
C                                                                          000270
F                            $OLD DO 10 LOOP                               000280
F                            $SUBR GI                                      000290
M 0,0,1,1,1,5000                 $RMU                                      000300
V 0,0,2,1,5000,9,1,1                                                       000310
V 0,0,2,1,5000,12,2,1                                                      000320
V 0,0,3,1,5000,15,3,1                                                      000330
V 0,0,2,1,5000,9,2,1                                                       000340
V 0,0,3,1,5000,15,3,1                                                      000350
V 0,0,2,1,5000,12,3,1                                                      000360
V 0,0,3,1,5000,15,3,1                                                      000370
V 0,0,3,1,5000,15,3,1                                                      000380
V 0,0,2,1,5000,12,3,1                                                      000390
V 0,0,3,1,5000,15,3,1                                                      000400
V 0,0,2,1,5000,9,3,1                                                       000410
V 0,0,3,1,5000,15,3,1                                                      000420
R 3                                                                        000430
V 0,0,2,1,5000,12,3,1                                                      000440
C                                                                          000450
V 0,0,2,1,5000,9,2,2             $END OF GI                                000460
R 3                                                                        000470
V 0,0,2,1,5000,12,3,1            $DISX ETC                                 000480
V 0,0,3,1,5000,15,4,1                                                      000490
V 0,0,3,1,5000,9,1,1                                                       000500
C                                                                          000510
V 0,1,0,1,5000,20,1,2            $A/RHO                                    000520
V 1,0,1,1,5000,18,3,1                                                      000530
F                             $SUBR DIAGON                                 000540
R 4                                                                        000550
V 0,0,2,1,5000,15,2,1                                                      000560
C                                                                          000570
V 0,1,2,1,5000,9,4,2                                                       000580
V 1,0,2,1,5000,9,2,2                                                       000590
R 3                                                                        000600
V 0,0,3,1,5000,15,4,1                                                      000610
V 0,0,3,1,5000,15,3,1                                                      000620
V 0,0,2,1,5000,9,3,1                                                       000630
C                                                                          000640
V 0,0,3,1,5000,15,3,1                                                      000650
V 0,0,2,1,5000,15,3,1                                                      000660
V 0,0,2,1,5000,9,3,1                                                       000670
F                             $END OF DIAGON                               000680
F                             $10 CONTINUE                                 000690
R 6                                                                        000700
V 0,0,2,1,100,9,3,1              $FFU=                                     000710
C                                                                          000720
V 0,0,3,1,100,15,3,1                                                       000730
V 0,0,2,1,100,9,1,2                                                        000740
F                             $13 CONTINUE                                 000750
F                             $VLYI - SOLVER PORTION                       000770
F                             $DOLLAR = JLEN/2                             000780
F                             $SINGLE QUOTE = KLEN                         000790
F                             $COLON = KLEN*JLEN/2                         000800
```

```
R 7                        ;CALL VTRI2 (7 TIMES)                000810
M 0,0,1,1,1,100                                                 000820
M 0,0,1,1,1,100                                                 000830
R 10                        ;DO J=JS1,JE1                       000840
V 0,1,2,1,100,9,3,1                                             000850
V 1,2,2,1,100,9,1,1                                             000860
V 2,3,0,1,100,20,1,2                                            000870
V 3,4,1,1,100,18,3,1                                            000880
V 4,0,2,1,100,15,2,2                                            000890
V 0,0,3,1,100,15,3,1                                            000900
C                                                               000920
R 10                        ;DO J=JS1,JE1                       000930
F                                                               000940
V 0,0,2,1,100,9,3,1                                             000950
C                                                               000960
C                          ;END OF VTRI2                        000970
V 0,0,2,1,100,9,2,2                                             000980
V 0,0,1,1,100,9,1,1                                             000990
M 0,0,1,1,1,100                                                 001000
M 0,0,1,1,1,100                                                 001010
R 3                        ;OLD DO 13 LOOP                      001020
V 0,0,2,1,5000,15,2,1                                           001030
V 0,0,3,1,5000,15,3,1                                           001040
V 0,0,1,1,5000,9,1,1                                            001050
V 0,0,2,1,5000,9,2,1                                            001060
C                                                               001070
V 0,0,3,1,5000,15,3,1                                           001080
V 0,0,1,1,5000,9,2,1          ;13 CONTINUE                      001090
R 2                        ;OLD DO 14 LOOP                      001100
V 0,0,3,1,5000,15,2,1                                           001110
V 0,0,3,1,5000,15,3,1                                           001120
V 0,0,3,1,5000,15,2,1                                           001130
V 0,0,1,1,5000,9,1,1                                            001140
C                                                               001150
V 0,0,3,1,5000,15,3,1                                           001160
V 0,0,3,1,5000,15,2,1                                           001170
V 0,0,1,1,5000,9,1,1                                            001180
V 0,0,3,1,5000,15,3,1                                           001190
V 0,0,3,1,5000,15,4,1                                           001200
V 0,0,3,1,5000,15,2,1                                           001210
V 0,0,3,1,5000,15,2,1         ;14 CONTINUE                      001220
F                          ;A3*DY1                              001240
F                          ;-CRKNIS                             001250
R 3                                                             001260
V 0,1,3,1,100,15,3,1        ;RMU*DY1*(UI-UI)                    001270
V 1,0,2,1,100,12,2,1        ;GUPK2=(A-B)*C  ETC                 001280
C                                                               001290
R 2                                                             001300
V 0,1,2,1,100,9,2,1         ;GU=T*GV                            001310
V 1,0,2,1,100,9,2,1         ;SBC=SBC+A*B  ETC                   001320
C                                                               001330
V 0,0,1,1,100,9,2,1         ;SBC(I,4)=A+B                       001340
F                          ;FORTH*DY1                           001350
V 0,1,3,1,100,15,3,1        ;(VSQ-VSQ)*C+FUSQ                   001360
V 1,0,3,1,100,15,4,1        ;USQ=USQ+T+FVSQ                     001370
V 0,1,3,1,100,15,3,1        ;DY1*RMU*(WSQ-WSQ)                  001380
V 0,0,3,1,100,15,3,1        ;DY1*RK*(EII-EII)                   001390
F                          ;-COSTSQ                             001400
V 1,2,3,1,100,15,3,1        ;RMU*(-COSTSQ)*T+S                  001410
V 2,3,3,1,100,15,4,1        ;A+FWSQ-B+F                         001420
V 3,0,2,1,100,9,2,1         ;SBC(I,5)=A+B*C                     001430
F                          ;JADD=                               001440
F                                                               001450
V 0,1,0,1,5000,20,1,2                                           001460
V 1,2,1,1,5000,18,3,1        ; RHOI=1/PRDICT                    001470
R 3                                                             001480
V 2,0,1,1,5000,9,2,1         ;U=P*RHOI  ETC                     001490
C                                                               001500
V 0,1,3,1,5000,15,2,1        ;U**2+V**2                         001510
V 1,2,3,1,5000,15,2,1        ;=.5*(T+W**2)                      001520
V 2,3,2,1,5000,9,3,1         ;EI=P*RHO+T                        001530
V 3,0,2,1,5000,15,2,1        ;P=A*B*C                           001540
```

```
F                         ;IF(JS1.LE.2)                                001550
M 0,0,1,1,1,100            ;P(1)=P(2)                                  001560
R 3                                                                    001570
M 0,0,1,1,1,100            ;U=U  ETC                                   001580
C                                                                      001590
V 0,0,1,1,100,9,1,1        ;V=-V                                       001600
F                         ;IF(I.LT.ILE) GO TO 253                      001610
V 0,0,2,1,100,9,2,2        ;W=-W $U=-U                                 001620
F                         ;IF(IADBWL.EQ.0)                             001630
F                         ;IF(I.NE.IE) GO TO 255                       001640
R 4                                                                    001650
M 0,0,1,1,1,5000            ;U=U ETC                                   001660
C                                                                      001670
R 10                                                                   001680
R 11                                                                   001690
F                                                                      001700
C                                                                      001710
C                         ;256 CONTINUE                                001720
R 2                                                                    001730
V 0,0,2,1,5000,9,2,1         ;UP=A+B*C ETC                             001740
C                                                                      001750
F                         ;IF(I.NE.IE) GO TO 70                        001760
R 2                                                                    001770
V 0,0,2,1,5000,9,2,1         ;UP=A+B*C ETC                             001780
C                                                                      001790
C                         ;70 CONTINUE -END OF N=1,2 LOOP              001800
R 5                                                                    001810
M 0,0,1,1,1,5000             ;RHOX=PRDICT                              001820
C                                                                      001830
C                         ;1 CONTINUE -END OF I=2,IE LOOP              001840
F                                                                      001850
F                                                                      001860
R 5                                                                    001870
M 0,0,1,1,1,100             ; RHO=RHO  ETC                             001880
C                                                                      001890
R 10                                                                   001900
R 5                                                                    001910
F                         ;B.C. AT K=KL                                001920
C                                                                      001930
C                                                                      001940
E                                                                      001950
```

```
            ****************************************
            *                                      *
            *      FLOATING POINT SAVEVALUES       *
            *                                      *
            ****************************************


NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
VECFP   1.0262380171560E+000     VECBZ   9.3682614968100E-001
CLKPD                 140233


NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
MAPFP   3.2089436825900E-005     MAPBZ   7.2137053984600E-002
```

```
H 32768,16,3,16          FFT ROUTINE                                         000099
R 8                         DO 15 L=1,16,2                                   000150
V 0,0,2,1,1344,9,2,2          RLD(L)=RLD(L)+RLD(M); RLD(M)=RLD(L)-RLD(M)     000190
V 0,0,2,1,1344,9,2,2          YMD(L)=YMD(L)+YMD(M); YMD(M)=YMD(L)-YMD(M)     000200
C                           15 CONTINUE                                      000210
R 3                         DO 89 I                                          000260
R 2                         DO 89 J=1,2(APPROX.)                             000330
V 0,0,2,1,1344,9,2,2          RLD(L); RLD(M)                                 000390
V 0,0,2,1,1344,9,2,2          YMD(L); YMD(M)                                 000400
R 2                         DO 89 K=2,3(APPROX)                              000440
V 0,0,3,1,1344,9,4,1          TI=YMD(M)*W(1,MM)-RLD(M)*W2(MM)                000500
V 0,0,3,1,1344,9,4,1          TR=RLD(M)*W(1,MM)+YMD(M)*W2(MM)                000510
V 0,0,2,1,1344,9,2,1          YMD(M); YMD(L)                                 000520
V 0,0,2,1,1344,9,2,1          RLD(M); RLD(L)                                 000530
C                                                                            000540
C                           89CONTINUE                                       000550
C                                                                            000560
R 2                                                                          000561
V 0,0,2,1,1344,9,2,2          LEFTOVERS FROM APPROX.                         000562
C                                                                            000563
R 8                                                                          000564
V 0,0,3,1,1344,9,4,1              "       "      "                           000565
V 0,0,2,1,1344,9,4,1              "       "      "                           000566
C                                                                            000567
V 0,0,2,1,1344,9,3,2          RLD0; YMD0                                     000570
E                                                                            000580
```

```
****************************************
*                                      *
*      FLOATING POINT SAVEVALUES       *
*                                      *
****************************************


NUMBER ............. CONTENTS    NUMBER ............. CONTENTS:
CLKPD                     161


NUMBER ............. CONTENTS    NUMBER ............. CONTENTS
VECFP    7.5786540768500E-001    VECBZ    9.8076699818000E-001
```

```
H 32768,16,3,16             LEGENDRE  TRANSFORM  ROUTINE  (SPCFOR)          000100
F   BEFORE THIS ROUTINE MAY BE UTILIZED, A SINGLE ELEMENT GATHER MUST       000110
F   BE DONE ONCE (AND ONLY ONCE FOR THE RUN OF THE ENTIRE MODEL) TO         000120
F   MAKE A MATRIX OF LEGENDRE COEFFICIENTS WHICH WILL BE WELL ORDERED       000130
F   AND CONFORMAL WITH THE MATRICIES IN THE FOLLOWING ROUTINE.              000140
F   THE GATHER IS:                                                          000150
F   DO 10 J=1,LR1            (1,23)                                         000160
F   DO 10 I=1,7                                                             000170
F   DO 10 K=1,NLATHF/2       (1,28)                                         000180
F   A(K,I,J)=P(I,J,K)                                                       000190
F   DO 10 M=1,NVERT          (1,48)                                         000200
F   TP(M,K,I,J)=A(K,I,J)                                                    000210
F   A TOTAL OF 10,000 SINGLE ELEMENT GATHERS (15000 CYCLES) -               000220
F   IN A TYPICAL WEATHER MODEL THIS SUBROUTINE IS CALLED SOME 75,000        000230
F   TIMES, MAKING THIS INITIAL INVESTMENT WELL WORTH THE OPPORTUNITY OF     000240
F   GETTING VECTOR LENGTHS OF 200 ELEMENTS.                                 000250
```

```
M 0,0,2,1,23,48          150  TASPEC(*,1,1:7)                                      000251
M 0,0,2,1,23,48          160  TASPEC(*,2,1:7)                                      000252
M 0,0,2,1,23,96   ??     170  TASPEC(*,3:4,1:7)                                    000253
M 0,1,2,1,23,192         180  TASPEC(*,5:8,1:7)                                    000254
V 1,0,3,1,1344,9,4,1         200  EVENR(*,1:8,L)                                   000260
V 0,0,3,1,1344,9,4,1         210  ODDR(*,1:8,L)                                    000270
V 0,0,2,1,1344,9,3,1         220  EVENR(*,1:8,L)                                   000280
V 0,0,2,1,1344,9,3,1         230  ODDR(*,1:8,L)                                    000290
V 0,0,2,1,1344,9,3,1         240  EVENR(*,1:8,L)                                   000300
V 0,0,2,1,1344,9,3,1         250  AGRID(*,1:8,L) & AGRID(*,9:16,L)                 000310
F                                                                                  000320
F                                                                                  000330
R 22                         DO 30 L=2,LR1                                         000340
M 0,0,2,1,11,48            310  TASPEC(*,1,ICE:ICE+11)                             000350
M 0,0,2,1,11,25            320  TASPEC(*,2,ICE:ICE+11)                             000360
M 0,0,2,1,11,96            330  TASPEC(*,3:4,ICE:ICE+11)                           000370
M 0,2,2,1,11,768          340  TASPEC(*,5:8,ICE:ICE+11)                            000380
M 3,0,1,1,1,400             470 & 490  AI(*,*,N+2-L)                               000381
F                                                                                  000390
V 2,0,3,1,1344,9,4,1         350  EVENR(*,*,L)                                     000400
V 0,0,3,1,1344,9,4,1         360  EVENI(*,*,L)                                     000410
V 0,0,3,1,1344,9,4,1         370  ODDR(*,*,L)                                      000420
V 0,0,3,1,1344,9,4,1         380 ODDI(*,*,L)                                       000430
V 0,0,2,1,1344,9,3,1         390  EVENR(*,*,L)                                     000440
V 0,0,2,1,1344,9,3,1         400  EVENI(*,*,L)                                     000450
V 0,0,2,1,1344,9,3,1         410  ODDR(*,*,L)                                      000460
V 0,0,2,1,1344,9,3,1         420  ODDI(*,*,L)                                      000470
V 0,3,2,1,1344,9,2,2         430 & 450  AGRID(*,1:8,L)!  AGRID(*,9:16,L)           000480
V 0,0,2,1,1344,9,2,2         440 & 460  AI(*,1:8,L) !  AI(*,9:16,L)                000490
V 0,0,2,1,1344,9,2,2         480 & 500 AI(*,1:8,N+2-L) ! AI(*,9:16,N+2-L)000500
C                           30   CONTINUE                                          000510
F                                                                                  000520
V 0,0,3,1,1344,9,4,1             550  EVENR(*,*,L)                                 000530
V 0,0,2,1,1344,9,3,1             560  EVENR(*,*,L)                                 000540
V 0,0,2,1,1344,9,3,1             570  AGRID(*,1:8,L)                               000550
F                                                                                  000560
V 0,0,3,1,29568,9,4,1             580  EVENR(*,*,2:LR1)                            000570
V 0,0,3,1,29568,9,4,1             600  EVENI(*,*,2:LR1)                            000580
V 0,4,2,1,29568,9,3,1             620  AGRID(*,1:8,2:LR1)                          000590
V 0,0,2,1,29568,9,3,1             640  AI(*,1:8,2:LR1)                             000600
R 22                         DO 79 L=2,LR1                                         000610
M 4,0,1,1,1,200             670  AGRID(*,1:8,N+2-L)                                000620
V 0,0,2,1,672,9,2,2         680  AI(*,1:8,N+2-L)                                   000630
C                           79   CONTINUE                                          000640
E                           RETURN                                                 000650


                ******************************************
                *                                        *
                *      FLOATING POINT SAVEVALUES         *
                *                                        *
                ******************************************


          NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
          VECFP    1.1388200815660E+000     VECBZ     9.5469946957600E-001
          CLKPD                   61590


          NUMBER ............. CONTENTS     NUMBER ............. CONTENTS!
          MAPFP    1.1203106020520E-003     MAPBZ     5.0836123116400E-001
```

```
H 32768,16,3,16       LINKHO  -  PARTIAL SIMULATION                                     000090
F                                                                                       000100
F            LINGHO ROUTINE                                                             000110
F                                                                                       000120
F                                                                                       000130
V 0,1,1,1,3456,9,2,1                                                                    000140
F                                                                                       000150
F                                                                                       000160
F                                                                                       000170
V 0,0,1,1,1152,9,1,1                                                                    000180
F                                                                                       000190
F                                                                                       000200
F                                                                                       000210
F                                                                                       000220
V 1,0,3,1,3456,15,4,1                                                                   000230
F                                                                                       000240
F                                                                                       000250
F                                                                                       000260
F                                                                                       000270
V 0,0,1,1,768,9,2,1                                                                     000280
F                                                                                       000290
F                                                                                       000300
F                                                                                       000310
F                                                                                       000320
V 0,0,1,1,384,9,2,1                                                                     000330
F                                                                                       000340
F                                                                                       000350
F                                                                                       000360
F                                                                                       000370
V 0,0,1,1,3456,9,2,1                                                                    000380
F                                                                                       000390
F                                                                                       000400
E                                                                                       000405
F                                                                                       000410
F                                                                                       000420
V 0,0,0,1,384,9,2,2                                                                     000430
F                                                                                       000440
F                                                                                       000450
V 0,0,0,1,384,9,1,1                                                                     000460
F                                                                                       000470
F                                                                                       000480
E                                                                                       000485
F                                                                                       000490
F                                                                                       000500
E                                                                                       000505
V 0,0,0,1,3456,20,2,2                                                                   000510
V 0,0,1,1,3456,18,3,1                                                                   000520
F                                                                                       000530
V 0,0,2,1,4608,15,2,1                                                                   000540
V 0,0,1,1,4608,20,2,2                                                                   000550
V 0,0,0,1,4608,18,3,1                                                                   000560
V 0,0,1,1,3456,20,2,2                                                                   000570
V 0,0,0,1,3456,18,3,1                                                                   000580
R 8                                                                                     000590
F                                                                                       000600
F                                                                                       000610
F                                                                                       000620
F                                                                                       000630
F                                                                                       000640
F                                                                                       000650
F                                                                                       000660
F                                                                                       000670
F                                                                                       000680
```

```
F                                                                                              000690
F                                                                                              000700
V 0,0,3,1,384,15,1,1                                                                           000710
F                                                                                              000720
F                                                                                              000730
F                                                                                              000740
F                                                                                              000750
F                                                                                              000760
V 0,0,3,1,384,15,2,1                                                                           000770
F                                                                                              000780
F                                                                                              000790
V 0,0,1,1,384,9,1,1                                                                            000800
V 0,0,1,1,384,20,2,2                                                                           000810
V 0,0,0,1,384,18,3,1                                                                           000820
C                                                                                              000830
F                                                                                              000840
F                                                                                              000850
F                                                                                              000860
V 0,0,2,1,3072,12,3,1                                                                          000870
F                                                                                              000880
V 0,0,3,1,3072,15,4,2                                                                          000890
V 0,0,2,1,3072,9,4,2                                                                           000900
F                                                                                              000910
F                                                                                              000920
V 0,0,1,1,3072,20,2,2                                                                          000930
V 0,0,0,1,3072,18,3,1                                                                          000940
F                                                                                              000950
F                                                                                              000960
F                                                                                              000970
F                                                                                              000980
V 0,0,2,1,384,12,3,1                                                                           000990
F                                                                                              001000
F                                                                                              001010
F                                                                                              001020
V 0,0,3,1,384,12,2,1                                                                           001030
V 0,0,2,1,384,12,3,1                                                                           001040
F                                                                                              001050
F                                                                                              001060
V 0,0,2,1,384,12,3,1                                                                           001070
E                                                                                              001080
```

```
                    ************************************
                    *                                  *
                    *     FLOATING POINT SAVEVALUES    *
                    *                                  *
                    ************************************


        NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
        VECFP    1.13457830912800E+000    VECBZ    9.71035489795000E-001



        NUMBER ............. CONTENTS     NUMBER ............. CONTENTS
        CLKPD                    16436
```

DIVISION 2

THE THREE-DIMENSIONAL AERODYNAMIC IMPLICIT CODE

DIVISION 2

THE THREE-DIMENSIONAL AERODYNAMIC IMPLICIT CODE

1.0 INTRODUCTION

A final recoding of the Ames implicit code has been completed with the proposed extensions under consideration for the FMP; these extensions are included in the FMP FORTRAN Manual (Volume III) and are discussed tutorially in Division 1 of this volume. This report presents coding strategy which was applied in recoding the implicit algorithm, to create an awareness of factors which affect performance of codes on the FMP. This is followed by a discussion of the recoding which was done. Performance analysis of this recoded version can be found in Division 1 of this volume.

2.0 CODING STRATEGY FOR THE 3-D IMPLICIT ALGORITHM

To achieve the optimum performance of the FMP, the programmer must be aware of the two key characteristics of the machine architecture:

a. Memory hierarchy
b. Functional parallelism

The data content of the larger production runs of the 3-D model make it necessary to allocate portions of the data base and working storage to each level of the memory hierarchy in the FMP. For example, a 100 x 100 x 100 mesh would entail the storage of 6 million flow variables (Q matrix), 3 million coordinate variables (X, Y, and Z), and 5 million intermediate results (S matrix). The expansion of this data into working storage areas for the block tridiagonal solution requires 25 x 3 x VL elements for the A, B, and C arrays plus 30 x VL elements for the L, U, and F arrays in the BTRI computation (VL=vector length). The engineering tradeoffs that have led to the current design of the FMP have dictated a maximum main memory configuration (using existing memory technologies) of 8 million 64-bit words. Decisions must be made as to where each of the major portions of the data in the implicit solution are to be retained. It does not seem feasible at this time for a compiler to be able to make the allocation determinations automatically, thus the programmer must use the LEVEL statements to advise the compiler (and the system) as to the desired storage assignments for each block of data.

This becomes even more necessary as one contemplates the creation of even larger "research" codes that have meshes of the order of tens of millions of points, since the third level of memory (LEVEL 3) which is made up of block-transfer-only Backing Storage is brought into play. The programmer must not only be aware of the relative storage capacities of each memory level, but also the implications in using that level of memory during processing. The basic groundrules are:

2.1 MEMORY HIERARCHY

2.1.1 MAIN MEMORY

a. Arithmetic memory-to-memory operations can only be performed from Main Memory.

b. Effective rates for arithmetic access in 64-bit mode of Main Memory are four sets of eight operands transferred to the Vector Units every machine clock cycle, and one set of 8 results stored back to memory in the same clock cycle.

c. In 32-bit mode these rates are doubled (four sets of 16 operands input, one set of 16 results).

d. Depending on the operation being performed, one, two, or three arithmetic processes (ADD, SUBTRACT, MULTIPLY) can be accomplished per set of operands delivered to a Vector Unit, per clock cycle.

e. Concurrent with vector arithmetic the Main Map Unit can achieve a simultaneous processing rate of 8 64-bit input operands per clock cycle, while storing 8 operands in the same clock cycle.

f. Single-element access rates to Main Memory for scalar load/store or vector scatter/gather operations are one per clock cycle.

2.1.2 INTERMEDIATE MEMORY

a. Data can be mapped in blocks to and from Intermediate Memory at a maximum rate of 8 elements every 3 clock cycles.

b. Single-word access rates are one element every 6 clock cycles.

c. No memory-to-memory arithmetic can be performed involving the Intermediate Memory.

d. Data transfers to and from Main Memory can proceed at the Intermediate Memory rates, and are fully concurrent with all Main Memory activities listed previously except other Main Map Unit operations.

e. The maximum configuration of Intermediate Memory is 32 million 64-bit words.

2.1.3 BACKING STORAGE

a. Data can only be transmitted between Backing Storage and Intermediate Memory.

b. Data can only be accessed and transferred in integral 32,768-word blocks.

c. Access time per block is negligible since a data transfer begins as soon as a starting address (current address within the block) is transmitted to the backing storage controller (Swap Unit).

d. Transfer rates for Backing Storage can attain a maximum of 8 64-bit words every 16 clock cycles.

2.2 FUNCTIONAL PARALLELISM

The memory system description above indicates the degree of concurrency available in the FMP. Maximum performance of the FMP is achieved by maximizing the overlap (or concurrency) of mapping operations (needed to organize data into efficient vectors) and the arithmetic processing. If one examines modern day scalar machines and FORTRAN object code, a "dual" can be found to this "scheduling" situation. Most high speed processors today possess the ability to engage in several simultaneous activities in order to attain high performance. In particular, the time required to access a data element in memory via a scalar load operation can be overlapped with the processing of other data that had been loaded previously. A great deal of work has been expended in compiler development to maximize the automatic scheduling of load, store, and arithmetic operations so that the arithmetic units are not left idle, while unnecessarily waiting on the results of a load operation to be returned from memory.

This same approach is used by the programmer and compiler for the FMP. Data for one set of arithmetic operations can be "prepared" by the Map Units while the Vector Units are operating on a previously aggregated set of data. The degree to which these processes can be overlapped determines the extent to which the Vector Units can be kept busy. The objective for maximum FMP performance would be to keep the Vector Units 100% active, performing triadic operations at every turn. This level of activity would yield an operation rate of 1.5 billion 64-bit floating-point operations per second, or 3 billion 32-bit floating-point operations per second. It is obvious that to attain a sustained rate of 1 billion floating-point operations per second in 64-bit mode, the combination of hardware, programmer, and compiler have to maintain a 67% efficiency in the use of the Vector Units. Given the current state of the art of compilers, it can be stated firmly that the programmer must provide some assistance in the statement of codes in order to achieve the requisite efficiency.

The coding strategy for the 3-D implicit code consists of the following general principles:

a. Allocation of all flow variables, coordinate arrays, and the intermediate (S) array to Intermediate Memory.

b. Reserving Main Memory for working storage and temporary holding areas for data being mapped to and from the Intermediate Memory.

c. Backing Storage usage to be invisible for this set of metrics, that is, relying on the Operating System to roll the entire job in and out of Backing Storage but no explicit data transfers during program execution.

d. Processing of "slabs", "chunks", or "pencils" of the data base at each step of the algorithm to maximize the vector lengths seen by the vector arithmetic operations.

e. Slab sizes to be limited by the available workspace in Main Memory.

f. Smaller problems can be run entirely in Main Memory as a single slab but the main program remains unchanged, provided declarations such as LEVEL are done dynamically.

g. When slabs or subsets of the major arrays are to be processed, they are explicitly described with subarray notation so that this usage is obvious to the reader and the compiler. This means that although the compiler may be able to discern a slab process from the construct

```
DO 10 J=1,JMAX
DO 10 L=L1,L2
DO 10 K=1,KMAX 10
RJ=Q(K,L,6,J)
```

the programmer should use the explicit notation

```
RJ=Q(*,L:LSL,6,*)
```

which highlights the fact that a slab of Q is being used.

This notation not only makes it clearer to the reader (particularly in a complex DO loop of several hundred lines) of the code but the compiler can deal with this single statement as a single map function. Note the duality of this concept; in scalar mode the statement at 10 would result in a scalar load, while the subarray statement generates a map operation. Both the load and the map may be handled by the compiler in similar ways, in terms of scheduling the resulting object code for efficient execution.

h. Special-casing of subroutines is used rather than single, general-purpose routines. For example, the XXM, YYM, and ZZM subroutines contain data dependent branch operations whose purpose is discernable at the time the program is being created. An in-line expansion of these routines in the STEP subroutine, eliminates the need for these branches, since during STEP the execution of XXM, YYX, and ZZM is not data dependent.

i. Solution algorithms and methodology (other than the use of slabs) remains unchanged.

j. As much as possible a line-by-line congruency is maintained between the original scalar coding and the FMP vectorized version. The major exceptions to this are the in-line incorporation of XXM, YYM, and ZZM, and the use of explicit notation for data structuring, restructuring, and transformation. The heavy use of the DEFINE and DYNAMIC FORTRAN extensions makes it possible to deal with the familiar scalar temporaries such as L11, L12,..., and RJ,RR,U,.... as temporary vectors (or arrays).

3.0 THE STEP SUBROUTINE

The most computationally intensive portion of the implicit code is found in STEP and the called subroutine BTRI. This set of programs has therefore received the most attention in developing language extensions and compiling strategies. The problem restatement will be examined as it impacts data movement and arithmetic in each of the three sweep directions.

3.1 SLABS

The maximum vector processing is achieved by ensuring vector lengths of more than 1000 elements so that the effect of vector startup time is negligible. It is desirable to reiterate here that since there are effectively 8 Vector Units (four units, half-clocked), a vector of length 8 would be processed in one clock cycle, but with a startup time of six to nine clock cycles. The non-arithmetic overhead of such an operation would be 600%-900% of the arithmetic processing time. At vector lengths of 1000 or greater this overhead amounts to .6%-.9%, or less, of the arithmetic processing time.

The first problem then, is to systematically divide the mesh to be processed into chunks that can be fed to the arithmetic unit as long vectors. Obviously, if all flow variable could be held in Main Memory, the entire mesh could be processed as a single vector. For a cubic mesh of N x N x N dimensions, the data storage required would be

$$14*N**3+105*N**2 \text{ elements}$$

where:

> N cubed is the number of mesh points, 14 the number of variables to be stored per mesh point, 105 the number of temporary variables per vector element, and N squared is the vector length (one plane of the cube).

Assuming an 8 million-word limit on contiguous high speed storage, then

$$14*N**3+105*N**2\leq 8000000$$

and thus N can be approximately 80 and there will still be space left for incidental working storage for the solution process.

In such a case all memory map oprations would proceed at a rate no slower than one element per clock cycle, and with a peak rate of 8 elements per clock cycle.

For meshes larger then 80x80x80 however, the computational variables must be held in Intermediate Memory, rather than Main Memory because of the storage requirements for data and temporaries. In this case any map operations involving Intermediate Memory would run slower than their counterparts in Main Memory. For example:

- Main Memory to Main Memory map operation, 8 64-bit words per clock
- Intermediate Memory to Main Memory map operation, 8 64-bit words per 3 clocks
- Intermediate Memory to Intermediate Memory operation, 4 64-bit words per clock.

In estimating performance, the problem arises as to what mix of data and memory should be used. That is, how much data should be stored in Main Memory and how much in Intermediate Memory? If for example, a mesh size of 85x85x85 were to be processed, all the flow variables could be retained in Main Memory and the X, Y, and Z matrices placed in Intermediate Memory. Then a process would be coded for 'slabbing' of the X, Y, and Z from Intermediate to Main Memory as the computations proceeded. As the mesh dimensions change the amount of inter-memory mapping also changes, and thus the performance rates change.

To simplify the estimation process, it was decided to assume that all meshes are held in Intermediate Memory regardless of size. If the resulting simulations show that the performance of the FMP on these meshes is at least one gigaflop in 64-bit mode, it is obvious that the same problem, if held in Main Memory only, would run at least as fast.

To provide maximum overlap, space must be allocated in Main Memory not only for the slab being processed, but also the next slab of data being mapped in from Intermediate Memory. Thus the execution of line 930 (from the FORTRAN listing found in appendix B, Division 1)

```
RJ=Q(2:KMAX-1,L:L+LSL-1,6,*)
```

would initite a map operation into a data area called RJ during its first pass. As soon as all flow variables have been mapped into Main Memory for the current value of L, code generated by the complier would perform the operation

```
RJ'=Q(2:KMAX-1,L+LSL:L+(2*LSL)-1,6,*)
```

where RJ' is a data buffer area created by the compiler and invisible to the programmer. This map operation would be carried out during the arithmetic processing of the first batch of mapped data. At the next pass through the loop, the pointers to data area RJ would be modified (without programmer intervention) by the object code to point to the new data area RJ'. Likewise, the pointers to RJ' would be modified to point to the original data area RJ.

This activity is similar to the technique used by multiregister machines and compilers that "prefetch" data into working registers during one trip through a DO loop, and move the data to a new register for processing during the next trip through the loop. This "invisible" allocation does impact the total main memory storage requirement however, and is a function of the slab size. Inversely, the slab size is a function of the available memory, the three mesh dimensions, and the sweep direction. The allocation of slab space is now examined for the J-sweep direction.

First a slab is made consisting of an integral number of planes (each consisting of JMAX*KMAX elements) as seen in figure 1. Vector lengths then are JMAX*KMAX*LSL elements where LSL is the number of planes in the L direction. Since working storage for the block tridiagonal is 105*(vector length), 105*JMAX*KMAX*LSL words of storage is required for temporary data.

Also, 14*KMAX*JMAX*LSL words of storage must be provided for the flow variables mapped to and from Intermediate Memory, plus an equal amount of storage for buffering the data for the next trip through the loop. The gross requirements for Main Memory are then

105*JMAX*KMAX*LSL+28*JMAX*KMAX*LSL=133*JMAX*KMAX*LSL

If for example, JMAX=KMAX=100, then 1,330,000 words of data would be required for each plane to be processed. Given a somewhat unintelligent and brute force data allocation to Main Memory, LSL would have to be about 6 in order to fit the problem within the 8 million words available in Main Memory. This would give vector lengths on the order of LSL*JMAX*KMAX=6*100*100=60,000 elements, which is remarkably close to the maximum size of 65,536 elements for the FMP. This optimizes vector performance by minimizing startup time per element in the vector.

The corresponding slab sizes for the other sweep directions, JSL and KSL are computed in the same manner. Note then that data allocation of arrays such as RJ is different in total size and dimensionality for each sweep direction (assuming JMAX,KMAX, and LMAX not identical). Thus such array variables necessarily become DYNAMIC elements in this code.

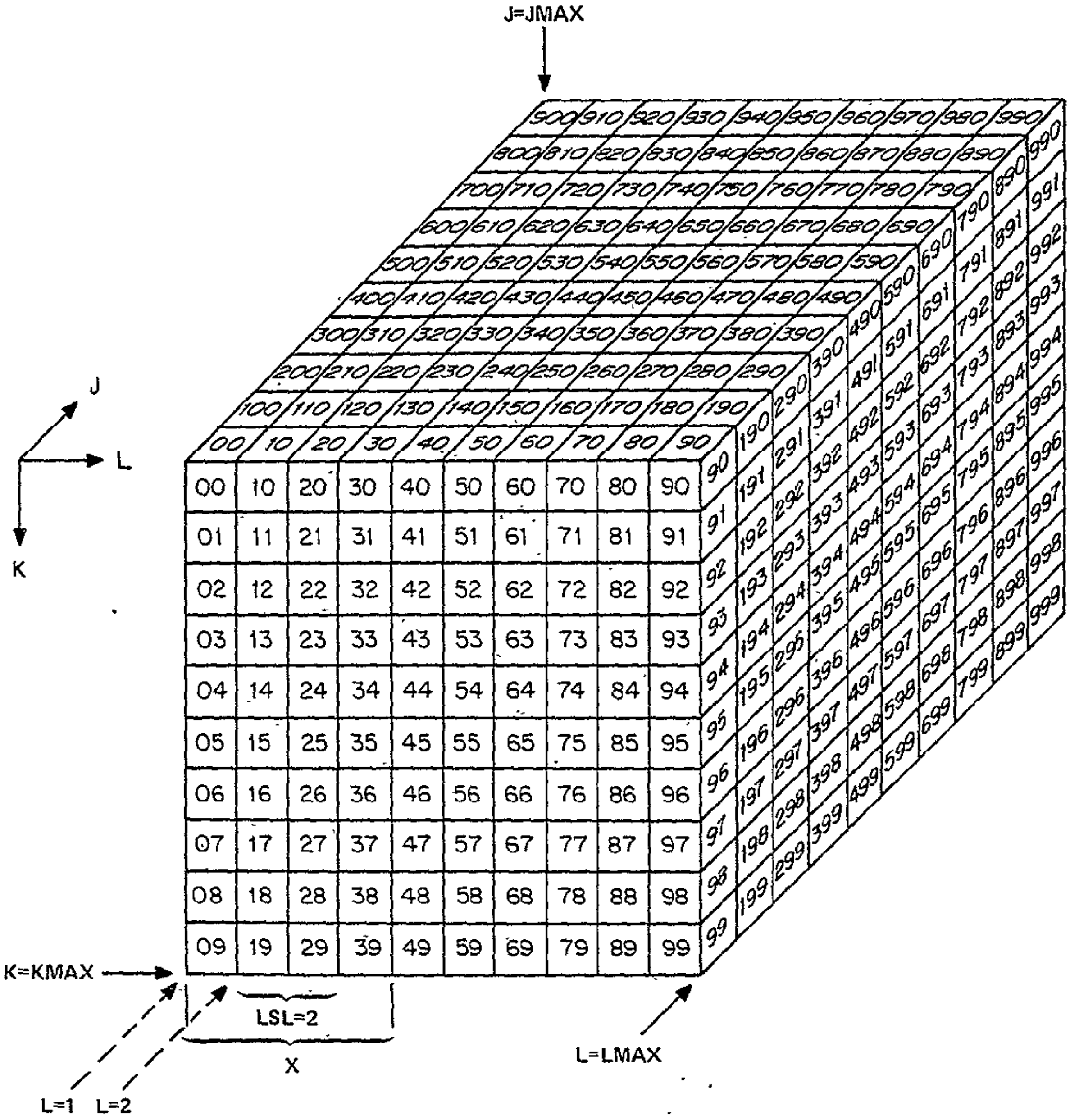


Figure 1. Storage Allocation for the X Array in Intermediate Memory

3.2 J-SWEEP DIRECTION

The first discussion will be of the processing for the left-hand-side solution, followed by the subroutines RHS, VISRHS, and MUTUR.

The computation of the residue (lines 500 through 680) has been left intact, and in place. The compiler is capable of automatically vectorizing this segment of code. However, the processing rate is limited by the rate at which the S matrix elements can be transmitted from Intermediate Memory (hereafter called LEVEL 2). A better approach for this computation would be to embed the RESIDUE calculation within SMOOTH, where S array elements are mapped into Main Memory.

Processing of the flow variables in the J-sweep direction begins at line 730. Figure 1 gives the storage mapping of a specimen X array of dimension 10 x 10 x 10 to demonstrate the method of "slabbing" the data. Elements of the X array are stored sequentialy in physical memory from 00 to 999. To improve vectorization the metric computation XXM is recoded in-line from lines 930 through 1100. Three intermediate variables not found in the original XXM routine (XKL,YKL, and ZKL) are created to hold the mapped X, Y, and Z slabs. This becomes necessary in order to minimize the number of times map operations are performed, as well as to provide the buffer space which holds mesh data for the differencing operations (lines 970 through 1020).

The statements

```
XKL=X(*,L-1:L+LSL,2:JMAX-1)
YKL=Y(*,L-1:L+LSL,2:JMAX-1)
ZKL=Z(*,L-1:L+LSL,2:JMAX-1)
```

produce map operations which are of the "prefetch and buffer" type described previously. This means that there are actually two buffer areas for each variable XKL, YKL, and ZKL, one of each "invisible" to the programmer. Note that the slab being moved for these variables is LSL+2 planes in size. This is due to the need for the adjacent differencing used in the metric computation. If LSL is 6, thus requiring 17 trips through the loop (16 trips for LSL planes and 1 trip for 4 planes), then 2 extra planes of data will be moved at each trip through the loop. The effect of this is moving 2*17+LMAX planes of data.

This extraneous movement could be reduced by explicitly programming the retention of the LSL+2 plane before commencing the next trip through the loop. This would require additional thought and data movement in the existing program. Instead, the brute force approach of moving the extra data has been taken to keep the program as similar to its scalar counterpart at possible. The overlapping of map operations ensures that this extra data motion is hidden by the computations being done in the loop. The only penalty for this technique then becomes the initial overhead in getting the first slabs of data ready for the first trip through the loop. In the worst case this additional burden becomes 2*KMAX*JMAX=2000 elements which are moved at a rate of 8 per 48-nanosecond period, or 12 microseconds per sweep per trip (time step) through the entire STEP subroutine.

Although a 10 x 10 x 10 mesh could fit in Main Memory, those dimensions will be used for this part of the discussion while referring to the FORTRAN listing (appendix B, Division 1) for the metric. Thus some of the figures which follow can be used to aid the illustration of the coding of the J-sweep. Therefore:

```
JMAX,KMAX,LMAX=10 LSL=2
```

Figure 2 shows the storage of the XKL array after executing the statement at line 940:

```
XKL=X(*,L-1:L+LSL,2:JMAX-1)
```

This statement becomes a map operation which can be called Gather Record wherein LSL+2 columns of KMAX operands are gathered JMAX times. The resulting data is moved to Main Memory and appears as the slab in figure 2. Note that the dimension of this temporary mesh is KMAX*JMAX*(2+LSL).

In statement 970

```
XK=(XKL(3:KMAX,2:LSL+1,*)-XKL(1:KMAX-2,2:LSL+1,*))*DY2
```

the entire slab is processed by a diadic arithmetic operation. The vector length for this operation is JMAX*(KMAX-2)*LSL elements. The compiler produces object code which computes the proper starting addresses for the offsets needed to compute the adjacent differences. A subsequent map operation is also generated which compresses the K=1 and K=KMAX elements from the array. This operation proceeds in parallel with the arithmetic statement 980. The result is shown in figure 3 where the final slab of dimension (KMAX-2)*LSL*JMAX is stored. From this point up to statement 1520 all slabs processed are conformal and have the same dimensions.

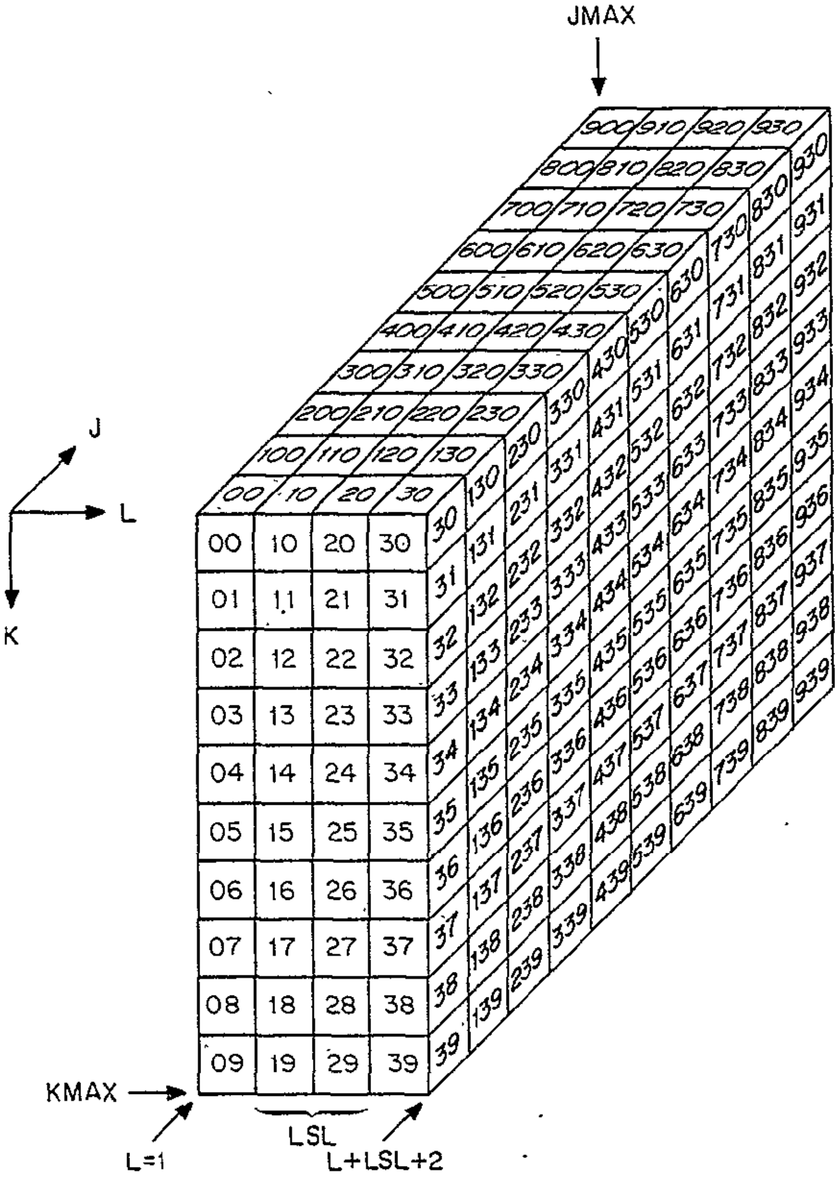


Figure 2. The J-sweep XKL, YKL, and ZKL Arrays in Main Memory

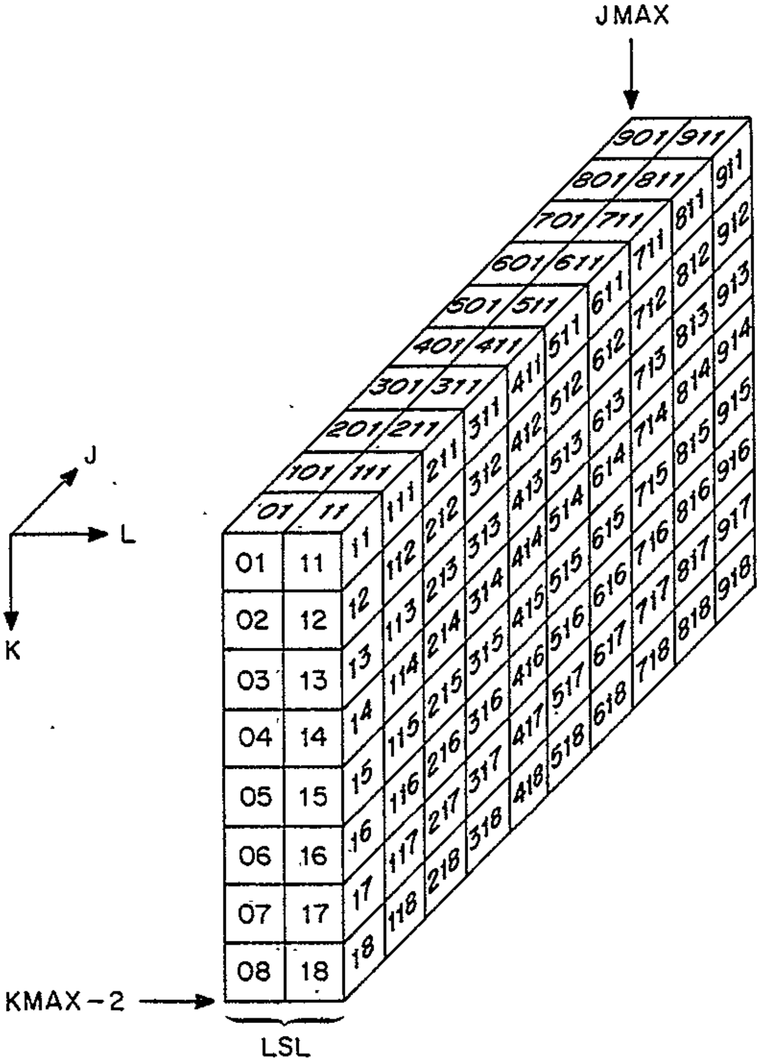


Figure 3. The J-sweep XK, YK, and ZK Arrays in Main Memory

Statements 930,1140 through 1170, and 1210 perform gather operations on operands in the Q matrix. Figure 4 shows the storage layout of the Q matrix in physical memory. Note that element 600 of the array is the first element in the J=2 segment of the Q array. This is due to the storage allocation that makes Q(K,L,1,J) the density component of the flow variables at the point (K,L,J) in the mesh. Thus the statement (line 930)

```
RJ=Q(2:KMAX-1,L:L+LSM,6,*)
```

results in a map operation that gathers KMAX-2 elements from each column for LSL columns, JMAX times. Figure 5 shows the resulting slab as it would appear in Main Memory, with each element labled with its original sequential storage location within the Q array in LEVEL 2 memory. Thus the first element moved from the Q matrix would be

```
RJ(1)=Q(2,2,6,1)
```

or element 511 in the linear storage of the array. Note that the entire slab can be processed as a single vector by all statements down to statement 1520.

The effect of some of the FORTRAN extensions on this sequence of code is now examined briefly. Statements 930 through 1020 make use of the explicit subarray notation described in the extension specification to define not only the slab to be processed but also the offsets

```
(3:KMAX...
(1:KMAX-2...
....3:LSL...
....1:LSL-2....
```

necessary for the adjacent differencing operation. The variables RJ, XKL, YKL, ZKL, XK, YK, ZK, XL, YL, and ZL are defined as DYNAMIC. Thus they take on the dimensionality of the right-hand expression, and their storage is allocated dynamically at object time.

Statements 1030 through 1060 then compute with these DYNAMIC variables producing arrays for each of the DYNAMIC ARRAY variables XX(1), XX(2), XX(3), and XX(4). In this case the DYNAMIC ARRAY XX will contain 4 sets of pointer information, with dimensionality established at object time and storage allocation also defined during execution of the statements. The DYNAMIC ARRAY variable D(i,j) is handled in much the same way.

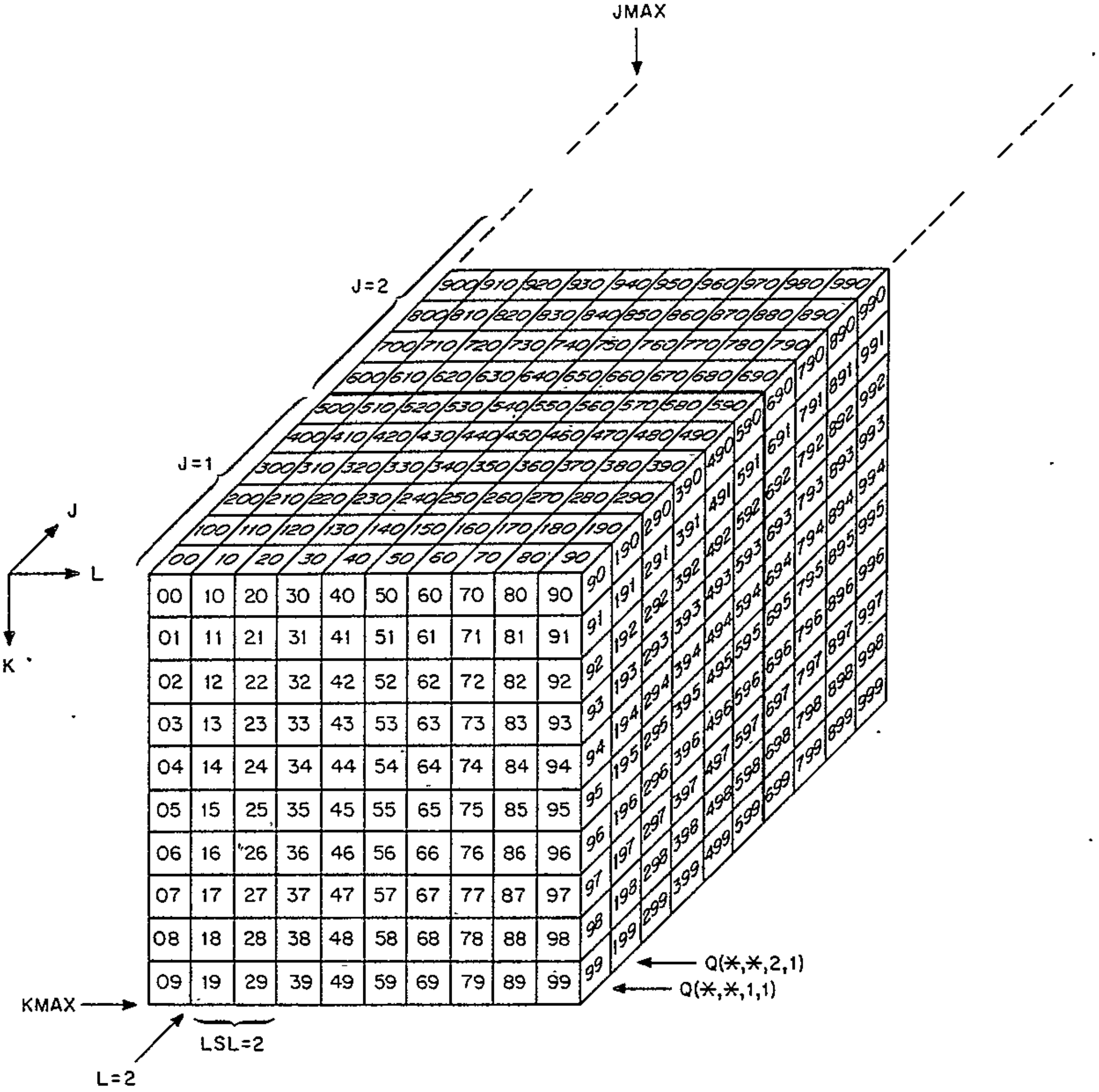


Figure 4. Q Matrix

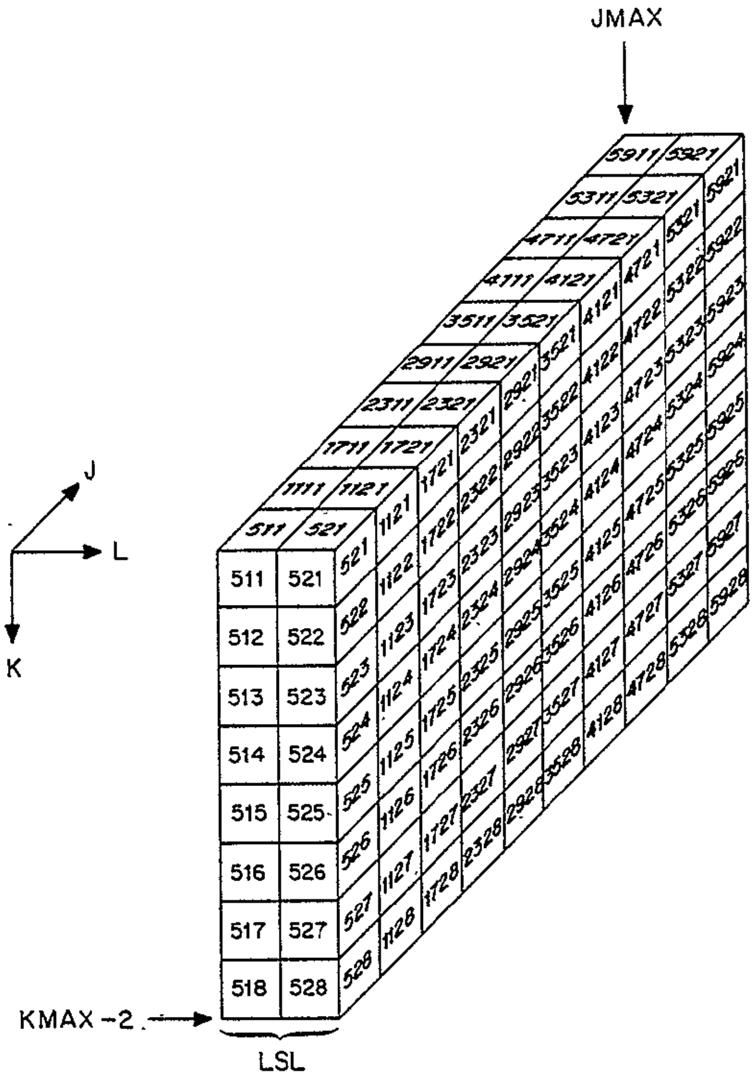


Figure 5. The J-sweep RJ Matrix in Main Memory
RJ=Q(2:KMAX-1,L:L+LSL-1,6,*)

At statement 1270 it is necessary to insert an explicit DEFINE statement

```
DEFINE (D(1,5),(1:KMAX-2,1:LSL-2,1:JMAX-2))
```

for although all other pointer data in the DYNAMIC ARRAY D is implicitly defined by the associated arithmetic statements such as

```
D(1,2)=XX(1)*HDX
```

where XX(1) is a slab (from figure 5) of dimensions (KMAX-2), LSL, and (JMAX-2), then D(1,2) takes on these identical characteristics. D(1,5) however, has not been given any such characteristics, and the desire is to fill this variable with scalar zeroes. The DEFINE statement explicitly establishes the necessary relationships and then statement 1280

```
D(1,5)=0
```

performs the filling of a created slab with the needed zeroes.

Statements 1520, 1530, and 1540 perform arithmetic operations on the slab called RJ. By simply adjusting the starting address and field lengths at object time, these operations yield new slabs with dimensions (KMAX-2)*LSL*(JMAX-2) without the need for any intervening map operations. The DO loops commencing at 1550 and continuing through 1650 are a slight restatement of the original scalar loops. This restatement is partly for convenience and expedience, and partly to make visible the power of the DYNAMIC ARRAY and DEFINE constructs in the FORTRAN extensions.

The purpose of statement 1570 is the same as 1270, to assign a set of attributes to all DYNAMIC ARRAY elements B(n,m), since the dimensionality is not established implicitly.

The purpose of statement 1580

```
DEFINE (D1,D(N,M))
```

is to establish the characteristics of a single DYNAMIC variable D1 so that they can be modified with offsets in statement 1590:

```
A(N,M)=-D1(*,*,1:JMAX-2)
```

This is necessary since DYNAMIC POINTERS appearing in assignment statements such as 1590 cannot have any modifiers included, but DYNAMIC VARIABLES such as D1 can possess a modified set of subscript notation such as shown here.

The result of these loops is to establish 25 pointers in the DYNAMIC ARRAYS A, B, and C which point to vectors of data (KMAX-2)*LSL elements in length. The BTRI subroutine then operates on these vectors as if each were in fact a scalar quantity in the original implicit code. In this manner the BTRI algorithm can be kept intact as provided by Ames.

Skipping to BTRI briefly (line 4120) this setting up of the input arrays is discussed.

3.3 BTRI

The DYNAMIC ARRAYS (pointer data) A, B, and C are passed to BTRI through COMMON. Notice that for every pointer in A a set of attributes exists which describe the vectors created in STEP. Each vector possesses three dimensions (K, L, J). At the beginning of the L-U decomposition the J=1 element should be extracted from each vector. This is done as in statement 4300

```
     DEFINE(B1(N,M),B(N,M)(*,*,1))
```

where a dummy DYNAMIC ARRAY B1 is created whose pointers each point to the J=1 element of the corresponding B(N,M). Note that this is the only construct wherein DYNAMIC ARRAY pointers can be redefined. The vector length of operations upon B(N,M) would be (KMAX-2)*LSL*(JMAX-2) elements whereas vector operations on B1(N,M) would involve lengths of (KMAX-2)*LSL elements! This is the effective length of most vector operations in the balance of the BTRI routine.

This technique is found again at statement 4870 and beyond....

```
     DEFINE (A1(N,M),A(N,M)(*,*,I))
```

In all of the computations in BTRI no map operations are required, just the recomputation of pointers and lengths at object time.

Other than the operation on vectors rather than scalars, BTRI appears almost identical to its scalar version in the original code.

On return from BTRI the elements of the S matrix are updated with the tridiagonal solution (see statement 1730)

```
     DO 24 N=1,5
     S1(N)=F1(N)
24   CONTINUE
```

where the pointer data in the DYNAMIC ARRAYS S1 and F1 were established in statements 870 and 880:

```
     DO 5 N=1,5
     DEFINE (F1(N),F(2:KMAX-1,L:L+LSL-1,N,2:JMAX-1))
5    DEFINE (S1(N),S(2:KMAX-1,L:L+LSL-1,N,2:JMAX-1))
```

3.4 K AND L SWEEPS

The memory mapping of arrays during the J-sweep have been diagramed in what may seem excessive detail. This was done to show the basic organization of data that is efficient for processing by the heavily compute-bound BTRI subroutine. Since vectors are linearly stored, the statements

```
     DIMENSION A(100,100,100) A(*,*,1)=2*A(*,*,1)
```

would result in the generation of a vector multiply operation of length 100x100, beginning at J=1. The statement

```
     A(*,1,*)=2*A(*,1,*)
```

would result in a series of J=100 vector operations, each of length 100 and

```
     A(1,*,*,)=2*A(1,*,*)
```

would result in a series of 100x100 scalar operations for all K1.

It can thus be seen for subscript notations of

```
     A(K,L,J)
```

that processes applied one at a time to each J index would yield vectors of length KMAX*LMAX. The J-sweep direction can therefore be vectorized with ease since the "natural" vectors are already stored in physical memory in the most efficient way.

The K-sweep is not an efficient method, since vectors are stored sequentially beginning with K=1, K=2...., K=KMAX, then for L=2 they continue in storage for K=1,..., etc. In the K-sweep direction however, the interest is not in vectors in the K direction, but instead for each K, a vector of length at least LMAX*JMAX is desired. To achieve this a method must be contrived to "transpose" the data from the flow and coordinate meshes so that the LMAX*JMAX vectors result. Using figure 1 to represent the original LEVEL 2 storage of the X matrix, figure 6 then shows the desired storage of the XJL array (slab) in Main Memory.

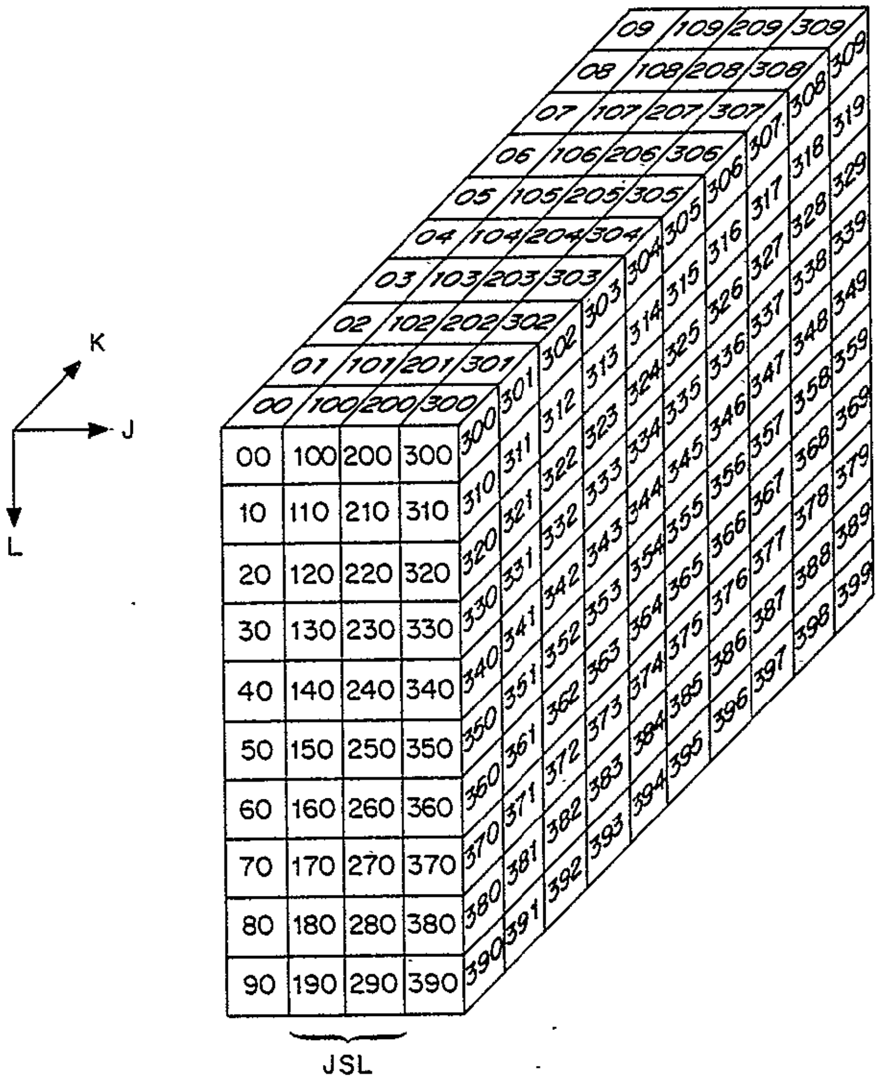


Figure 6. The K-sweep Matrix XJL in Main Memory

Following the previous representation of physically sequential storage of data by vertical column, with each vertical column contiguous by plane, figure 6 shows the new positions of X array elements as they would appear in Main Memory. Each block in figure 6 contains the original sequential number of the data as it appeared in LEVEL 2 memory. Note that subscript axis (upper left-hand corner of the figure) shows the new directions implied by the subscripts K, L, and J.

To achieve this transposition, the map operations of data from LEVEL 2 memory must be described differently than were coded in the first (J-sweep) phase of this code. The transposition is explicitly described by statements 1900 through 2020, wherein RJ, XJL, YJL, ZJL, and the transposed arrays Q1 through Q5 are created. Taking the XJL map operation from statement 1920

```
XJL(1:LMAX,1:JSL+1,K)=X(K,1:LMAX,J-1:J+JSL)
```

it can be seen that for every trip through the loop, K will be advanced by one element. The statement shown becomes then:

> "For each K, transfer a vector of length LMAX*(JSL+2) to the array XJL".

The map operation called for can be accomplished with a single map function which will not only move the requisite vector but will perform the move for KMAX times. Note that this map operation must make single element references to LEVEL 2 storage for each element of the vector in the L and J directions. This requires 6 clock cycles per reference instead of the 3/8 of a cycle used for the corresponding map operations in the J-sweep direction. The key to the FMP achieving its performance goals is tied directly to the ability of the compiler to overlap this "slow transpose" with computations. A rough estimate suffices to show that for STEP this overlap is achievable in principle:

Elements to be moved from LEVEL 2 per mesh point:

a. Q1 through Q5 = 5 elements
b. X, Y, Z = 3 elements
c. Additional data for adjacent differencing = 2/LSL= 2/6=1/3 element
d. S1 through S5 = 5 elements
e. Update of S1 through S5 = 5 elements

TOTAL------------------18.3 elements per grid point

Six clock cycles are required per element moved = 6*18.3 = 110 clock cycles. The peak 64-bit arithmetic rate = 24 results per clock cycle; thus in 110 clock cycles the Vector Unit could produce 24*110 = 2640 results maximum, or 880 results minimum. To achieve complete overlap of the mapping functions necessary in the K-sweep direction, then at least 880, or at most 2640, different arithmetic operations would have to be performed on each grid point of the whole mesh. The heart of the BTRI subroutine itself yields many more arithmetic operations than

2640, so that if the compiler can properly schedule the respective map operations, almost all transpose operations can be overlapped.

Once the transpose operations are completed (by the time statement 2090 is reached) the resulting main memory slabs can be proccessed identically by AMATRX, FILTRY, FILTRZ, and BTRI functions as were used in the J-sweep direction. These functions are retained as in-line code to maintain congruency with the original scalar code, as well as to permit the compiler to perform more intelligent scheduling of the map operations.

The L-sweep direction is handled in exactly the same manner as the K-sweep with a transpose operation called out at the beginning of the loop and a transpose operation required for the updating of the Q matrix in LEVEL 2 storage after computations are complete. Figure 7 shows the corresponding storage of the XLJ array in Main Memory for this sweep direction. Note again the revised subscript directions in the upper left corner of the figure. As in the K-sweep direction, once the transpose has been completed the computations on the main memory slabs are identical to those in the J- and K-sweep portions of the STEP code.

Two major differences should be noted however. One is the retention of the originally mapped Q array data in a main memory buffer until it is updated in statements 3980 through 4080. The other is the retention of the Q1 through Q5, D(1,1) through D(1,4), and ZZ(1) through ZZ(4) temporary arrays for use in VISMAT so that no additional map operations are required in that routine.

The transpose operation in the L-sweep direction proceeds at the same rate as the simple map operations in the J-sweep portion of the program. At a maximum, 8 elements are moved every 3 clock cycles (see description of gather in section 2.1.3, Division 1, this volume). This is 16 times faster than the single-element transfers required for the K-sweep direction. Thus only 165 arithmetic operations are required to ensure overlap of processing in the Vector Unit and Map Unit for those sweeps. This number of vector operations is much less than the nearly 4000 operations that appear in each sweep direction in STEP.

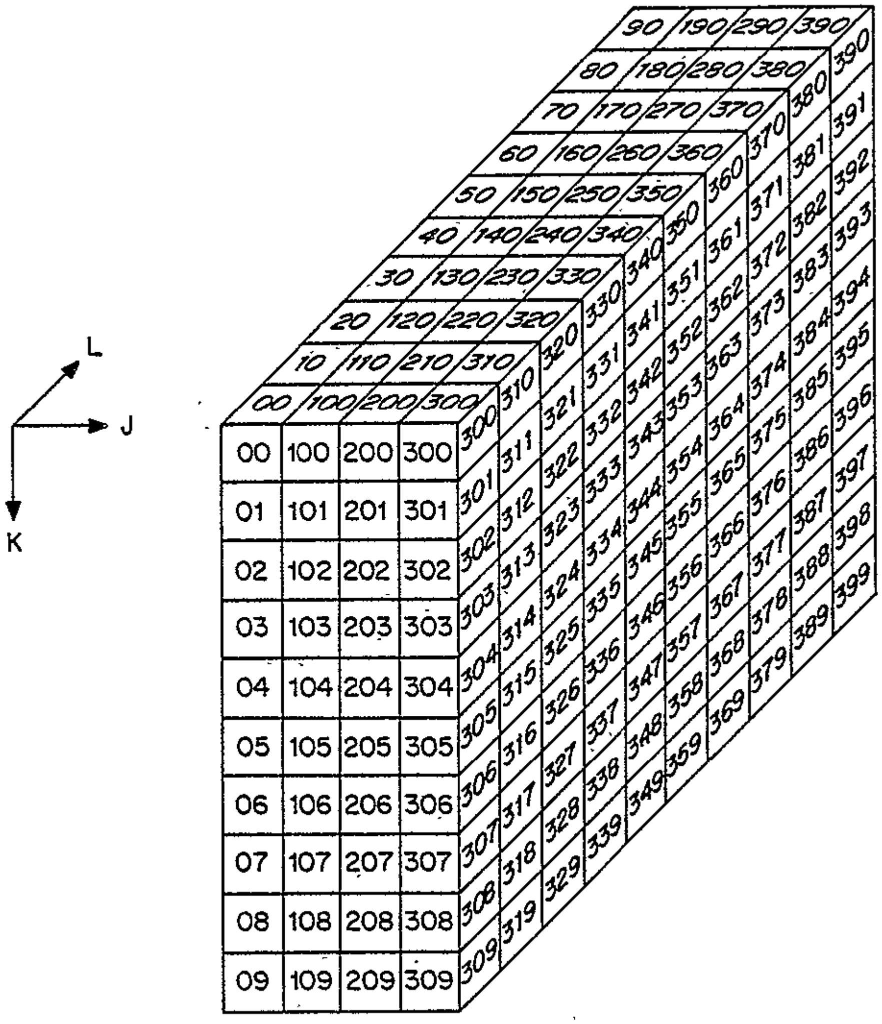


Figure 7. The L-sweep Matrix XLJ in Main Memory

4.0 THE RIGHT-HAND SIDE COMPUTATION

The "slab" technique used in the left-hand side computation is not as efficient in machine usage when employed for the right-hand side (RHS). The major reason for this is that the amount of arithmetic computation required for the right-hand side is substantially less than that needed in the left-hand solution, and data mapping for the right-hand side becomes the governing commodity in the calculation rate.

As an example, assume that the throughput rate for the right-hand side is limited to the map rate for the data to be used for computations. The data to be moved would be

a. 3 X, Y, Z quantities
b. 5 Q mesh quantities
c. 5 S matrix quantities (moved back to Intermediate Memory)

for the first (L direction) pass. In order to provide the data for the adjacent differencing needed in metric calculations, 2 additional planes of data must be moved for each slab. Thus if a slab consists of 6 planes, then 2/6 more data is moved than is required for the final computation. This extra data is moved from the X, Y, and Z meshes, so that 3*2/6 = 1 more element of data on the average needs to be moved per mesh point. In the first pass then, 3 + 6 + 5 + 1 = 15 elements are moved per mesh point calculated on. The time needed to move this data from Intermediate Memory is

15*3/8 clock cycles = 6 clock cycles per mesh point.

For the second pass (K direction) the 15 data elements already described must be moved plus retrieving the intermediate S mesh elements for a total of 15 + 5 = 20 elements. The rate for this move is the single-element map rate of one element every 6 cycles. Thus the per-point cost of mapping is

20*6 = 120 clock cycles.

The third pass (J direction) requires the transfer of 20 elements at the gather-record rate of 3/8 elements per clock cycle. The cost of this map per mesh point then is

20*3/8 = 8 clock cycles.

The total number of cycles committed to data mapping for the right-hand side (excluding viscosity calculations) is

6 + 120 + 8 = 134 clock cycles per mesh point.

A quick count of the floating-point operations for the right-hand side shows that a total of 221 operations are performed per mesh point. If 64-bit mode only is assumed, the rates for the Vector Units are

1 arithmetic operation = 8 ops per clock cycle = .5 gigaflop,

2 arithmetic operations = 16 ops per clock cycle = 1 gigaflop,

3 arithmetic operations = 24 ops per clock cycle = 1.5 gigaflops.

The number of clock cycles needed to process the 221 operations required per mesh point for the various gigaflop rates are

221/8 = 28 cycles for a .5 gigaflop rate,
221/16 = 14 cycles for a 1 gigaflop rate,
221/24 = 10 cycles for a 1.5 gigaflop rate.

Note that it takes 134 cycles just to move the data, but only 28 cycles (at worst) to compute with it. The right-hand side is thus Map Unit bound. Thus the gigaflop rate for this processing method is

GIGAFLOP RATE = NUMBER OF OPS/(NUMBER OF CYCLES*16)

for a 16-nanosecond clock cycle.

RATE = 221/(134*16) = .103 gigaflops.

This is clearly much less than the 1 gigaflop rate objective for the FMP, and even when combined with the left-hand side computation, the RHS acts as a major constraint on the implicit code performance.

The best solution to this dilemma is to recode the right-hand side to reduce the amount of data mapping necessary. The key to this approach is that there is no recursive relationship between the data used at each pass. Therefore it is possible to map a complete slab from Intermediate Memory (see figure 8) and process all three sweeps for that slab. Not only does this reduce the number of data transfers, but the single-element gather operation is eliminated.

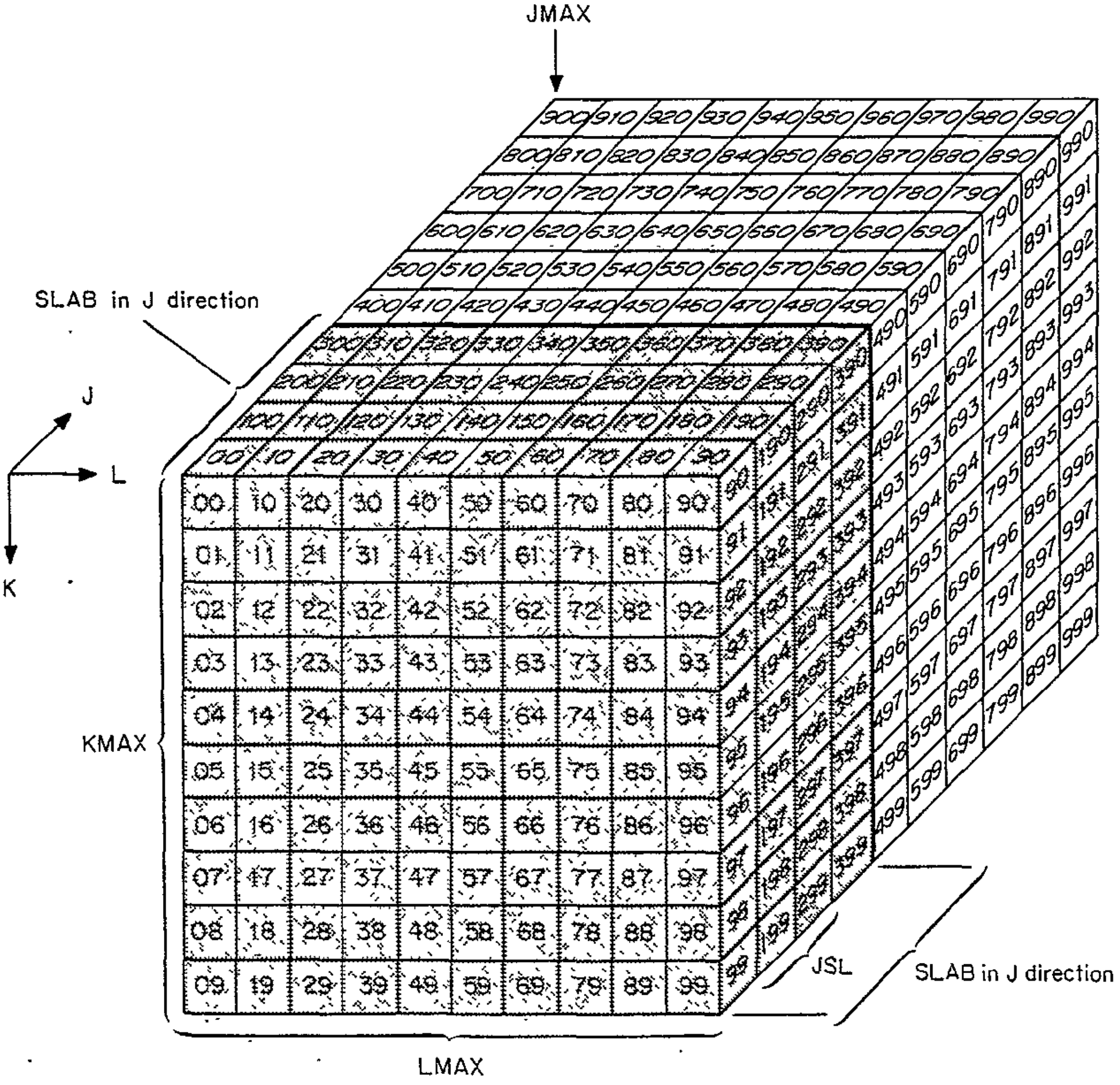


Figure 8. Right-hand Side X Array in Intermediate Memory - Matrix XJL Shown Before Move to Main Memory

Lines 260 through 1430 of the RHS listing (found in appendix C, Division 1) give the major DO loop wherein the slabs are moved along the J direction. Note that 9 elements are moved to Main Memory and the slab of S array is retained there until all updates are complete, then mapped back to LEVEL 2 memory at the completion of each pass through the RH loop. To reduce map operations the choice was made to process all data for points at K = 1, K = KMAX and L = 1, L = LMAX even though the operations are meaningless. Thus in a 100x100x100 mesh this method will have passed over 40,000 extra data elements that need not be processed. This is an added burden of 4% on the number of computations against a benefit of more straightforward programming which results in fewer (slower) map operations. Note that when the S matrix is mapped back to LEVEL 2 memory, the unneeded data points at K = 1, KMAX and L = 1, LMAX are discarded.

The computation rate for this approach is:

9 elements mapped into Main Memory
5 S elements returned to LEVEL 2 memory
1 element per mesh point moved as an extra for adjacent differencing
15 elements at 8/3 element moved per cycle = 6 cycles per mesh point

A glance at the arithmetic rates given previously for the FMP Vector Units shows that if a way can be found to achieve the maximum of 1.5 gigaflops, 10 clock cycles per mesh point will be required for computation. Obviously the 6-cycle map time can be completely overlapped by the computation (assuming a smart compiler can schedule the object code).

5.0 RIGHT-HAND SIDE--VISCOSITY AND TURBULENCE COMPUTATIONS

The right-hand side calculations are complicated somewhat by the inclusion of two subroutine calls which have been retained in the recoding for the FMP--VISRHS and MUTUR.

The restructuring of this code needed for the FMP consists of bringing the large slabs from Intermediate Memory in the most efficient form possible (minimal memory conflicts during the gather operation).

5.1 VISRHS

With the storage algorithm chosen for the flow variables and X,Y,Z data (the L sweep direction proceeds along the FORTRAN-stored "rows" of data), this means that the maximum length of vectors for many calculations in VISRHS are no more than column length of the flow mesh -- KMAX elements.

To improve the processing rate of the FMP for this computation, VISRHS (viscosity calculations) contains a transpose operation on the U, V, and W arrays that were generated in the RHS subroutine. The time required for the three key transpose operations is offset by the ability to process vectors of KMAX*JSL*LMAX elements. With no recursion in VISRHS, this technique permits maximum vector performance for the FMP.

If the parameter LAMIN is not equal to zero, the turbulence model MUTUR is called. This routine offers some programming of interest since there are both explicit recursion and data dependent IF branches involved in that routine.

5.2 MUTUR

The computations found in MUTUR are a "mixed bag" for the effective use of the FMP. It was decided that transposition operations on all the necessary meshes in MUTUR--ZZ,XX,YY,RR,Q1...Q6--was too expensive to undertake for the performance returned in the vector arithmetic operations. Thus the meshes were left in their original orientation. This means that vector lengths will be as small as KMAX elements. The first approach to conversion consisted of changing the program into a rational form for the STAR compiler. This involved the replacement of the three-dimensional array references with four-dimensional array references. The use of the compound variable KL is thus eliminated. Appendix A is a listing of the original MUTUR code but with KL eliminated.

Once this variable was removed the intent of the code is more obvious to the reader and to the compiler. Evaluation of the resulting program by an FMP FORTRAN compiler could possibly yield optimal results. However, to make clear the probable restructuring done by either programmer or the compiler, the choice was made to recode the problem using extensions provided by the proposed FMP FORTRAN.

C-4

The next step was to recode as many statements as possible in the subarray form. It became obvious that the subscripts and loops at the beginning of MUTUR could be replaced by an "unrolled" form found in lines 430 through 990 of appendix B, the recoded MUTUR. These statements yield vectors of length KMAX*LMAX*JSL, despite the fact that there are offsets in the subscripts of +1 and -1 for all three dimensions. A feature of the FMP is its ability to discard results under the guidance of a bit-by-bit control vector. In this case the compiler is capable of discerning the fact that Q3(3:KMAX+2,*,*) will involve only adjustment of starting addresses and discarding of results later. Thus the length of the operation will be KMAX*LMAX*JSL although the useful results will, in the final output, be (KMAX-2)*(LMAX-2)*JSL elements.

The presence of IF statements in original FORTRAN source code usually causes some difficulty in the vectorization process. In MUTUR there were four different classes of IF statements, each of which has to be resolved by analyzing the <u>intent</u> of the code.

1) The first class of IF statement in MUTUR was apparently used to reduce source and object code computation of arrays TA, TB, TC.... through manipulation of subscripts and a one-time looping back to recompute a new set of values. This was resolved, as discussed previously, by unrolling the loop into statements 430 through 990. Although unrolling of the loop requires a greater allocation of object code, the intent of that sequence seems to be clearer in the restructured version (see appendix B).

2) The second class of IF statement is used to identify and locate the maximum and minimum elements of arrays UU and YDU. The replacement construct shown in lines 1260 and 1270

   ```
        DO 21 L=2,LM
   21   IF(UU (*,L,*).LT.UMIN(*,1,*))UMIN(*,1,*)=UU(*,L,*)
   ```

   creates object code wherein a control vector is generated for every J,K element at the position L in the matrix, with a one representing the fact that the particular K,L,J element of UU is less than the corresponding element in UMIN. The entire statement 21 then causes a replacement (using the Map Unit) of all UMIN elements by new minimum values where indicated by the control vector. A slight modification in the source statement would have instead generated the machine instruction Q8SMIN (minimum of vector X):

   ```
        DO 21 L=2,LM
   21   IF(UU (*,L,J).LT.UMIN(*,1,J))UMIN(*,1,J)=UU(*,L,J)
   ```

   where the maximum would be evaluated for vectors of length KMAX. If the position of the minimum element is desired, the programmer must invoke the 'Q8 IN-LINE' construct that permits direct access to the hardware from the FORTRAN compiler. Thus the previous example would become

```
      DO 21 L=2,LM
21    CALL Q8SMIN(UU(*,L,J),UMIN,MINPOS)
```

In this case the parameter MINPOS will be returned with the index of the minimum element in array UU.

Examination of the MUTUR code showed that the maximum length for the minimum function under these circumstances would be KMAX elements. By using the control vector and processing planes of data, the maximum and minimum functions can also be accomplished, but at a much higher performance level.

In lines 1280 through 1330 the control vector is generated for a plane of the meshes at each L and then used to suppress or permit the movement or insertion of data into the arrays YDUM, YM, and KM. This control vector use is performed solely by the Map Unit, while the generation of the control vector requires the use of the Vector Units to accomplish the arithmetic compare operations.

Note that in line 1330 a single scalar is broadcast to all elements of KM2 where permitted by the control vector. The construct

```
      IF (BIT)......
```

where BIT is defined to be of type BIT, will normally generate control vector operations.

3) The third class of IF statement appears to have the same function as that of the second, that is, the selective insertion of data into meshes under the control of some conditions of interest. This class becomes converted to control vector operations as shown in statement 1490 through 1560 and also in statements 1570 and 1580 (although less obvious in this latter situation). A bit vector BIT is generated at line 1380. It is then combined in a logical operation with another bit vector generated at line 1490 resulting in a new BIT at line 1500. This new BIT is thence used to control the modification of corresponding elements of YMM and YDU in statements 1510 and 1520. Note that in this case the arithmetic is performed for all elements of YM, AM, BM, and YDUM but results are stored into the result vector only where permitted by the control vector. Since this is a data dependent operation, estimation of processing rates is difficult. In the analysis of the simulation results (see Division 1) this will be dealt with for MUTUR by providing three possibilities for the contents of the control vectors -- 0%, 50%, and 100% density of permissive bits. The processing rate range can then be determined for this portion of the code.

4) The final IF statement class is similar also to the last two, but has one additional area for caution and analysis. In this case (lines 1840 through 1860) elements of TURMU are

to be set depending upon the position of a crossover point in the values of TMO and TMI. Until the crossover point is reached data is to be moved from TMI, and after the crossover, data is to be moved from TMO. In this particular situation, analysis shows that all elements of TMO and TMI have values which are on the proper side of the crossover point. That is, elements 1 through n of TMI will all be less than their counterparts in TMO. If n is chosen as the crossover point then it can be said that all elements from n to LMAX of TMO will be less than those in TMI. It is this fact that allows simply replacing the IF statements with a control vector operation which will place the correct TMO and TMI elements in TURMU.

If, however, the positional relationship of TMO elements and TMI elements did not hold on both sides of the crossover point, a form of recursion exists which requires manipulating the control vectors to produce the proper 'mask' for movement of data.

Because of recursions like that in lines 1920 through 1950 it has been necessary to retain the DO loops in L and to process planes of data at each point L. The maximum vector length in all of the recursive portions is thus KMAX, although the compiler does generate all of the necessary vector operations in the J direction, automatically. No gather operations are indicated, as the Map Unit expense would overshadow the value that longer vectors in arithmetic might have.

APPENDIX A

ORIGINAL MUTUR ROUTINE

```
      SUBROUTINE MUTUR
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH
     1  ,DX1,DY1,DZ1,ND,ND2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ
     2,RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(720,6,30)
      COMMON/VAR0/S(720,5,30)
      COMMON/VAR1/X(720,30),Y(720,30),Z(720,30)
      COMMON /VAR3/P(120,30),XX(60,4),YY(60,4),ZZ(60,4)
      LEVEL 2,Q,S,X,Y,Z
      COMMON/COUNT/NC,NC1
      COMMON/BTRID/A(60,5,5),B(60,5,5),C(60,5,5),D(60,5,5),F(60,5)
      COMMON TURMU(720,30)
      DIMENSION TAS(60),UU(60),SNOR(60),TMO(60),TMI(60),S5(60),S6(60),
     1  U(60),V(60),W(60),E(60),RR(60)
      EQUIVALENCE (P(1,1)   ,TAS(1)),(P(1,2)   ,UU(1)),(P(1,3)   ,SNOR(1)),
     *(P(1,4)   ,TMO(1)),(P(1,5)   ,TMI(1)),(P(1,6)   ,S5(1)),
     *(P(1,7)   ,S6(1)),(P(1,8)   ,U(1)),(P(1,9)   ,V(1)),(P(1,10)  ,W(1)),
     *(P(1,11)  ,E(1)),(P(1,12)  ,RR(1))
      DATA F27,LEDGE/1.6,25/
      DATA FK,FKK,YDUMF/0.4,0.0168,1.0/
      DATA FKLEB/0.3/
C
C  CALCULATE TURBULENT VISCOSITY
C
      DO 80 J = 2,JM
      DO 80 K = 2,KM
C    ZZM COMPUTATIONS ELIMINATED,ZZ RETAINED FROM RHS CALCULATIONS
C
C  CALCULATE VORTICITY TAS(L) AND TOTAL VELOCITY UU(L)
C
      DO 11 L = 1,LM
      RL1 = 1./Q(K,L+1,1,J)
      RL = 1./Q(K,L,1,J)
      T1 = .5*((ZZ(L+1,2)+ZZ(L,2))*(Q(K,L+1,4,J)*RL1-Q(K,L,4,J)*RL)
     1  -(ZZ(L+1,3)+ZZ(L,3))*(Q(K,L+1,3,J)*RL1-Q(K,L,3,J)*RL))
      T2 = .5*((ZZ(L+1,3)+ZZ(L,3))*(Q(K,L+1,2,J)*RL1-Q(K,L,2,J)*RL)
     1  -(ZZ(L+1,1)+ZZ(L,1))*(Q(K,L+1,4,J)*RL1-Q(K,L,4,J)*RL))
      T3 = .5*((ZZ(L+1,1)+ZZ(L,1))*(Q(K,L+1,3,J)*RL1-Q(K,L,3,J)*RL)
     1  -(ZZ(L+1,2)+ZZ(L,2))*(Q(K,L+1,2,J)*RL1-Q(K,L,2,J)*RL))
      S1 = 0.0
      S2 = 0.0
      S3 = 0.0
C    XXM COMPUTATIONS ELIMINATED,DUE TO RETENTION OF XX RESULTS
C    YYM COMPUTATIONS ELIMINATED,YYM RETAINED FROM RHS CALCULATIONS
      RL1 = 1./Q(K,L,1,J+1)
      RL = 1./Q(K,L,1,J-1)
      TA = .5*XX(J,2)*(Q(K,L,4,J+1)*RL1-Q(K,L,4,J-1)*RL)
     1  -.5*XX(J,3)*(Q(K,L,3,J+1)*RL1-Q(K,L,3,J-1)*RL)
      TB = .5*XX(J,3)*(Q(K,L,2,J+1)*RL1-Q(K,L,2,J-1)*RL)
     1  -.5*XX(J,1)*(Q(K,L,4,J+1)*RL1-Q(K,L,4,J-1)*RL)
      TC = .5*XX(J,1)*(Q(K,L,3,J+1)*RL1-Q(K,L,3,J-1)*RL)
     1  -.5*XX(J,2)*(Q(K,L,2,J+1)*RL1-Q(K,L,2,J-1)*RL)
      RL1 = 1./Q(K+1,L,1,J)
      RL = 1./Q(K-1,L,1,J)
      TD = .5*YY(K,2)*(Q(K+1,L,4,J)*RL1-Q(K-1,L,4,J)*RL)
     1  -.5*YY(K,3)*(Q(K+1,L,3,J)*RL1-Q(K-1,L,3,J)*RL)
      TE = .5*YY(K,3)*(Q(K+1,L,2,J)*RL1-Q(K-1,L,2,J)*RL)
     1  -.5*YY(K,1)*(Q(K+1,L,4,J)*RL1-Q(K-1,L,4,J)*RL)
      TF = .5*YY(K,1)*(Q(K+1,L,3,J)*RL1-Q(K-1,L,3,J)*RL)
     1  -.5*YY(K,2)*(Q(K+1,L,2,J)*RL1-Q(K-1,L,2,J)*RL)
```

```
      S1 = S1+.5*(TA+TD)
      S2 = S2+.5*(TB+TE)
      S3 = S3+.5*(TC+TF)
      RL1 = 1./Q(K,L+1,1,J+1)
      RL = 1./Q(K,L+1,1,J-1)
      TA = .5*XX(J,2)*(Q(K,L+1,4,J+1)*RL1-Q(K,L+1,4,J-1)*RL)
     1  -.5*XX(J,3)*(Q(K,L+1,3,J+1)*RL1-Q(K,L+1,3,J-1)*RL)
      TB = .5*XX(J,3)*(Q(K,L+1,2,J+1)*RL1-Q(K,L+1,2,J-1)*RL)
     1  -.5*XX(J,1)*(Q(K,L+1,4,J+1)*RL1-Q(K,L+1,4,J-1)*RL)
      TC = .5*XX(J,1)*(Q(K,L+1,3,J+1)*RL1-Q(K,L+1,3,J-1)*RL)
     1  -.5*XX(J,2)*(Q(K,L+1,2,J+1)*RL1-Q(K,L+1,2,J-1)*RL)
      RL1 = 1./Q(K+1,L+1,1,J)
      RL = 1./Q(K-1,L+1,1,J)
      TD = .5*YY(K,2)*(Q(K+1,L+1,4,J)*RL1-Q(K-1,L+1,4,J)*RL)
     1  -.5*YY(K,3)*(Q(K+1,L+1,3,J)*RL1-Q(K-1,L+1,3,J)*RL).
      TE = .5*YY(K,3)*(Q(K+1,L+1,2,J)*RL1-Q(K-1,L+1,2,J)*RL)
     1  -.5*YY(K,1)*(Q(K+1,L+1,4,J)*RL1-Q(K-1,L+1,4,J)*RL)
      TF = .5*YY(K,1)*(Q(K+1,L+1,3,J)*RL1-Q(K-1,L+1,3,J)*RL)
     1  -.5*YY(K,2)*(Q(K+1,L+1,2,J)*RL1-Q(K-1,L+1,2,J)*RL)
      S1 = S1+.5*(TA+TD)
      S2 = S2+.5*(TB+TE)
      S3 = S3+.5*(TC+TF)
      TW = (T1+S1)**2+(T2+S2)**2+(T3+S3)**2
      TAS(L) = SQRT(TW)
      UTOT = Q(K,L,2,J)**2+Q(K,L,3,J)**2+Q(K,L,4,J)**2
      UU(L) = SQRT(UTOT)/Q(K,L,1,J)
      TURMU(K,L,J) = 0.0
   11 CONTINUE
C
C  COMPUTE RA
C
      L = 1
      WMU = 1.
      TAU = ABS(TAS(L))
      RA = SQRT(RE*Q(K,L,6,J)*Q(K,L,1,J)*TAU/WMU)/26.
C
C  COMPUTE NORMAL DISTANCF SNOR(L) AND YDUM
C
      SNOR(1) = 0.
      KM2 = 1
      YDUM = 1.E-3
      YM = .5/SQRT(ABS(ZZ(1,1)*ZZ(2,1)+ZZ(1,2)*ZZ(2,2)+ZZ(1,3)*ZZ(2,3)))
      UMAX = UU(1)
      UMIN = UU(1)
      YDUS = 0.0
      DO 20 L = 2,LM
      IF(UU(L) .LT. UMIN) UMIN = UU(L)
      SCIS = ABS(ZZ(L-1,1)*ZZ(L,1)+ZZ(L-1,2)*ZZ(L,2)+ZZ(L-1,3)*ZZ(L,3))
      SCAL = 1.0/SQRT(SCIS)
      SNOR(L) = SNOR(L-1) + SCAL
      SNORA = 0.5*(SNOR(L) + SNOR(L-1))
      YDU = SNORA*ABS(TAS(L-1))*(1.-EXP(-RA*SNORA))
      IF(L.GT.LEDGE)GO TO 20
      IF(UU(L).GT.UMAX) UMAX=UU(L)
      IF(YDU .LT. YDUM) GO TO 20
      KM2 = L - 1
      YDUM = YDU
      YM = SNORA
   20 CONTINUE
C
C     INTERPOLATE TO FIND YM, YDUM, AND UM
C
      IF(KM2 .LT. 2 .OR. KM2 .GT. LEDGE-1) GO TO 22
```

```
      YM3 = 0.5*(SNOR(KM2+1) + SNOR(KM2+2))
      YM1 = 0.5*(SNOR (KM2-1) + SNOR (KM2))
      YDUM1 = YM1*ABS(TAS(KM2-1))*(1.0 - EXP(-RA*YM1))
      YDUM3 = YM3*ABS(TAS(KM2+1))*(1.0 - EXP(-RA*YM3))
      C2 = YDUM - YDUM1
      C3 = YDUM3 - YDUM
      DY2 = YM - YM1
      DY3 = YM3 - YM
      AM = (DY3*DY3*C2+DY2*DY2*C3)/(DY2*DY3*(DY2+DY3))
      BM = (DY2*C3 - DY3*C2)/(DY2*DY3*(DY2 + DY3))
      IF(BM .GE. 0.) GO TO 22
      YMM = YM - 0.5*AM/BM
      YDU  = YDUM - 0.25*AM*AM/BM
      YMM = YM - 0.5*AM/BM
      IF(YDU .LT. YDUM .OR. YMM .LT. YM1 .OR. YMM .GT. YM3) GO TO 22
      YDUM = YDU
      YM = YMM
      IF(YM .GT. SNOR(KM2+1)) KM2 = KM2 + 1
      IF(YM .LT. SNOR(KM2)) KM2 = KM2 - 1
   22 CONTINUE
C
C      COMPUTE OUTER EDDY VISCOSITY
C
      DO 25 L=1,LEDGE
      SNOR(L) = 0.5*(SNOR(L) + SNOR(L+1))
      FFC = FKK*F27*RE*Q(K,L,1,J)*Q(K,L,6,J)
      TMO(L) = FFC*YM*YDUM
      FFCWK = YDUMF*YDUMF*FFC
      UDIFF = ABS(UMAX-UMIN)
      IF(YDUM .GT. UDIFF*YDUMF) TMO(L) = FFCWK*YM*UDIFF*UDIFF/YDUM
      FIA = FKLEB*SNOR(L)/YM
      IF(FIA .GT. 1.E5) FIA = 1.E5
      FI = 1.0 + 5.5*FIA**6
      TMO(L) = TMO(L)/FI
      TMO(L) = ABS(TMO(L))
   25 CONTINUE
C
C      COMPUTE INNNER EDDY VISCOSITY
C
      DO 30 L=1,LEDGE
      TAU = ABS(TAS(L))
      TMI(L) = Q(K,L,6,J)*Q(K,L,1,J)*RE*TAU*(.4*SNOR(L)*(1.-EXP(-RA
     1  *SNOR(L))))**2
      TMI(L) = ABS(TMI(L))
   30 CONTINUE
C
C      LOAD VISCOSITY COEFFS. INTO ARRAY, USE INNER VALUE UNTIL
C      MATCH POINT IS REACHED
C
      L = 1
   40 TURMU(K,L,J) = TMI(L)
      L = L+1
      IF(L.GT.LEDGE) GO TO 10
      IF( TMI(L) .LE. TMO(L)) GO TO 40
   41 TURMU(K,L,J) = TMO(L)
      L = L+1
      IF( L.LE. LEDGE) GO TO 41
   10 CONTINUE
C
C     REARRANGE TURMU(L) SUCH THAT WHEN AVERAGED AT L AND L+1
C        THE CORRECT MIDWAY VALUE WILL BE OBTAINED
C
      SMP = TURMU(K,1,J)
      DO 60 L = 2,LMAX
      TURMS = TURMU(K,L,J)
      TURMU(K,L,J) = 2.0*SMP-TURMU(K,L-1,J)
   60 SMP = TURMS
   80 CONTINUE
      RETURN
      END
```

APPENDIX B

RECODED MUTUR ROUTINE

```
      SUBROUTINE MUTUR
      COMMON/BASE/NMAX,JMAX,KMAX,LMAX,JM,KM,LM,DT,GAMMA,GAMI,SMU,FSMACH
     1  ,DX1,DY1,DZ1,ND,NO2,FV(5),FD(5),HD,ALP,GD,OMEGA,HDX,HDY,HDZ
     2,RM,CNBR,PI,ITR,INVISC,LAMIN,NP,INT1,INT2,INT3
      COMMON/GEO/NB1,NB2,RFRONT,RMAX,XR,XMAX,DRAD,DXC
      COMMON/READ/IREAD,IWRIT,NGRI
      COMMON/VIS/RE,PR,RMUE,RK
      COMMON/VARS/Q(720,6,30)
      COMMON/VAR0/S(720,5,30)
      COMMON/VAR1/X(720,30),Y(720,30),Z(720,30)
      COMMON /VAR3/ P, XX, YY, ZZ
      LEVEL 2,Q,S,X,Y,Z
      COMMON/COUNT/NC,NC1
      COMMON /BTRID/ A, B, C, D, F
      COMMON TURMU
      DYNAMIC P,XX,YY,ZZ,A,B,C,D,F,F1
      DYNAMIC TAS,TMO,TMI,S5,S6,U,V,W,E,RR
      DYNAMIC YDUM,YDUM1,YDUM3,YM,YM1,YM3,SCIS,SCAL
      DYNAMIC BIT,DY2,SY3,AM,BM,YMM,BIT1
      DYNAMIC ZZ1,ZZ2,ZZ3,ZZ4,ZZ5,RL1,RL
      DYNAMIC T1,T2,T3,S1,S2,S3,TA,TB,TC
      DYNAMIC TD,TE,TF,TW,TAS,UTOT,UU,TURMU
      DYNAMIC TAU,WMU,RA,SNOR,SNORA,
      DATA F27,LEDGE/1.6,25/
      DATA FK,FKK,YDUMF/0.4,0.0168,1.0/
      DATA FKLEB/0.3/
C
      DEFINE (TAS,P(1:60,1)),(UU,P(1:60,2)),(SNOR,P(1:60,3))
      DEFINE (TMO,P(1:60,4)),(TMI,P(1:60,5)),(S5,P(1:60,6))
      DEFINE (S6,P(1:60,7)),(U,P(1:60,8)),(V,P(1:60,9))
      DEFINE (W,P(1:60,10)),(E,P(1:60,11)),(RR,P(1:60,12))
C
C   CALCULATE TURBULENT VISCOSITY
C
      DEFINE (ZZ1,ZZT(1)),(ZZ2,ZZT(2)),(ZZ3,(ZZT(3)),(ZZ4,ZZT(4))
      DEFINE (ZZ5,ZZT(5))
C     ZZM COMPUTATIONS ELIMINATED,ZZ RETAINED FROM RHS CALCULATIONS
C
C   CALCULATE VORTICITY TAS(L) AND TOTAL VELOCITY UU(L)
C
      DEFINE (RL1,RR(*,1:LM,2:JSL)),(RL,RR(*,2:LMAX,2:JSL))
      T1 = .5*((ZZ2(*,2:LMAX)+ZZ2(*,1:LMAX,*))*(Q4(*,2:LMAX,*)*RL1
     1  -Q4(*,*,*)*RL)-(ZZ3(*,2:LMAX+1,*)+ZZ3(*,*,*))*(Q3(*,2:LMAX+1,*)
     2    *RL1-Q3(*,*,*))
      T2=.5*((ZZ3(*,2:LMAX+1,*)+ZZ3(*,*,*))*(Q2(*,2:LMAX+1,*)*RL1
     1   -Q2(*,*,*)*RL)-(ZZ1(*,2:LMAX+1,*)+ZZ1(*,*,*))
     2   *(Q4(*,2:LMAX+1,*)*RL1-Q4(*,*,*)*RL))
      T3=.5*((ZZ1(*,2:LMAX+1,*)+ZZ1(*,*,*))*(Q3(*,2:LMAX+1,*)*RL1
     1  -Q3(*,*,*)*RL)-(ZZ2(*,2:LMAX+1,*)+ZZ2(*,1:LMAX))
     2  *(Q2(*,2:LMAX+1,*)*RL1-Q2(*,*,*)*RL
      S1 = 0.0
      S2 = 0.0
      S3 = 0.0
C     XXM COMPUTATIONS ELIMINATED,DUE TO RETENTION OF XX RESULTS
C     YYM COMPUTATIONS ELIMINATED,YYM RETAINED FROM RHS CALCULATIONS
      DEFINE (RL1,RR(*,*,3:JSL+2),(RL,RR(*,*,1:JSL))
      TA = .5*XX(2)*(Q4(*,*,3:JSL+2)*RL1-Q4(*,*,*)*RL)
     1  -.5*XX(3)*(Q3(*,*,3:JSL+2)*RL1-Q3(*,*,*)*RL)
      TB = .5*XX(3)*(Q2(*,*,3:JSL+2)*RL1-Q2(*,*,*)*RL)
     1  -.5*XX(1)*(Q4(*,*,3:JSL+2)*RL1-Q4(*,*,*)*RL)
      TC = .5*XX(1)*(Q3(*,*,3:JSL+2)*RL1-Q3(*,*,*)*RL)
     1  -.5*XX(2)*(Q2(*,*,3:JSL+2)*RL1-Q2(*,*,*)*RL)
      DEFINE (RL1,RR(3:KMAX,*,*),RL(RR(1:KMAX-2,*,*))
```

```
      TD = .5*YY(2)*(Q4(3;KMAX+2,*,*)*RL1-Q4(*,*,*)*RL)
     1 ,-.5*YY(3)*(Q3(3;KMAX+2,*,*)*RL1-Q3(*,*,*)*RL)
      TE = .5*YY(3)*(Q2(3;KMAX+2,*,*)*RL1-Q2(*,*,*)*RL)
     1  -.5*YY(1)*(Q4(3;KMAX+2,*,*)*RL1-Q4(*,*,*)*RL)
      TF = .5*YY(1)*(Q3(3;KMAX+2,*,*)*RL1-Q3(*,*,*)*RL)
     1  -.5*YY(2)*(Q2(3;KMAX+2,*,*)*RL1-Q2(*,*,*)*RL)
      S1 = S1+.5*(TA+TD)
      S2 = S2+.5*(TB+TE)
      S3 = S3+.5*(TC+TF)
      DEFINE (RL1,RR(*,3;LM2,3;JSL+2),(RL,RR(*,3;LM2,*))
      KM2=KMAX+2
      LM2=LMAX+2
      TA = .5*XX(2)*(Q4(*,3;LM2,3;JSL+2)*RL1-Q4(*,3;LM2,*)*RL)
     1  -.5*XX(3)*(Q3(*,3;LM2,3;JSL+2)*RL1-Q3(*,3;LM2,*)*RL)
      TB = .5*XX(3)*(Q2(*,3;LM2,3;JSL+2)*RL1-Q2(*,3;LM2,*)*RL)
     1  -.5*XX(1)*(Q4(*,3;LM2,3;JSL+2)*RL1-Q4(*,3;LM2,*)*RL)
      TC = .5*XX(1)*(Q3(*,3;LM2,3;JSL+2)*RL1-Q3(*,3;LM2,*)*RL)
     1  -.5*XX(2)*(Q2(*,3;LM2,3;JSL+2)*RL1-Q2(*,3;LM2,*)*RL)
      DEFINE (RL1,RR(3;KM2,3;LM2,*)),(RL,RR(*,3;LM2,*))
      RL = 1./Q1(*,3;LM2,*)
      TD = .5*YY(2)*(Q4(3;KM2,3;LM2,*)*RL1-Q4(*,3;LM2,*)*RL)
     1  -.5*YY(3)*(Q3(3;KM2,3;LM2,*)*RL1-Q3(*,3;LM2,*)*RL)
      TE = .5*YY(3)*(Q2(3;KM2,3;LM2,*)*RL1-Q2(*,3;LM2,*)*RL)
     1  -.5*YY(1)*(Q4(3;KM2,3;LM2,*)*RL1-Q4(*,3;LM2,*)*RL)
      TF = .5*YY(1)*(Q3(3;KM2,3;LM2,*)*RL1-Q3(*,3;LM2,*)*RL)
     1  -.5*YY(2)*(Q2(3;KM2,3;LM2,*)*RL1-Q2(*,3;LM2,*)*RL)
      S1 = S1+.5*(TA+TD)
      S2 = S2+.5*(TB+TE)
      S3 = S3+.5*(TC+TF)
      TW = (T1+S1)**2+(T2+S2)**2+(T3+S3)**2
      TAS = SQRT(TW)
      UTOT = Q2**2+Q3**2+Q4**2
      UU = SQRT(UTOT)/Q1
      TURMU(*,*,J;JSL)=0
C
C   COMPUTE RA
C
      WMU = 1.
      TAU= ABS(TAS(*,1,*))
      RA = SQRT(RE*Q6(*,1,*)*Q1(*,1,*)*TAU/WMU)/26.
C
C   COMPUTE NORMAL DISTANCE SNOR(L) AND YDUM
C
      DEFINE (SNOR,(1;KMAX,1;LMAX,1;JSL)),(YDUM,(1;KMAX,1,1;JSL))
      DEFINE (SNORA,(1;KMAX,1;LMAX,1;JSL)),(YM,(1;KMAX,1,1;JSL))
      SNOR(1) = 0.
      KM2 = 1
      YDUM = 1.E-3
      YM = .5/SQRT(ABS(Z41(*,1,*)*ZZ1(*,2,*)+ZZ2(*,1,*)*ZZ2(*,2,*)
     1      +ZZ3(*,1,*)*Z43(*,2,*)))
      YDUS = 0.0
      SCIS=ABS(ZZ1(*,*,*)*ZZ1(*,2;LMAX+1)+ZZ2(*,*,*)*ZZ2(*,2;LMAX+1)
     1    +ZZ3(*,*,*)*ZZ3(*,2;LMAX+1,*))
      SCAL = 1.0/SQRT(SCIS)
      DO 19 L=2,LMAX-1
      SNOR(L) = SNOR(L-1) + SCAL
      SNORA = 0.5*(SNOR(L) + SNOR(L-1))
 19   CONTINUE
      YOU(*,2;LMAX+1,*)=SNORA(*,2;LMAX+1,*)*ABS(TAS))*
     1    *(1.EXP(-RA(*,2;LMAX+1,*)*SNORA(*,2;LMAX+2,*))
      DO 21 L=2,LM
 21   IF(UU(*,L,*).LT.UMIN) UMIN(*,1,*)=UU(*,L,*)
      DO 18 L=2,LEDGE
```

```
      IF(UU(*,L,*).GT.UMAX(*,1,*))UMAX(*,1,*)=UU(*,L,*)
      BIT=(YDU(*,L,*).LT.YDUM(*,1,*)
      IF(BIT)YDUM(*,1,*)=YDU(*,L,*)
      IF(BIT)YM(*,1,*)=SNORA(*,L,*)
      IF(BIT)KM2(*,L,*)=L-1
   18 CONTINUE
C
C     INTERPOLATE TO FIND YM, YDUM, AND UM
C
      BIT=(KM2(*,*,*).LT.2.OR.KM2(*,*,*).GT.LEDGE-1)
      YM3 = 0.5*(SNOR(KM2(*,*,*)+1) + SNOR(KM2(*,*,*)+2))
      YM1 = 0.5*(SNOR (KM2(*,*,*)-1) + SNOR (KM2(*,*,*)))
      YDUM1 = YM1*ABS(TAS(KM2(*,*,*)-1))*(1.0 - EXP(-RA*YM1))
      YDUM3 = YM3*ABS(TAS(KM2(*,*,*)+1))*(1.0 - EXP(-RA*YM3))
      C2 = YDUM - YDUM1
      C3 = YDUM3 - YDUM
      DY2 = YM - YM1
      DY3 = YM3 - YM
      AM = (DY3*DY3*C2+DY2*DY2*C3)/(DY2*DY3*(DY2+DY3))
      BM = (DY2*C3 - DY3*C2)/(DY2*DY3*(DY2 + DY3))
      BIT1=(BM.GE.0)
      BIT=(.NOT.BIT.OR..NOT.BM)
      IF(BIT)YMM = YM - 0.5*AM/BM
      IF(BIT)YDU = YDUM - 0.25*AM*AM/BM
      BIT1=(YDU.LT.YDUM.OR.YMM.LT.YM1.OR.YMM.GT.YM3)
      BIT=BIT.AND..NOT.BIT1
      IF(BIT)YDUM=YDU
      IF(BIT)YM=YMM
      IF(YM .GT. SNOR(KM2(*,*,*)+1)) KM2(*,*,*) = KM2(*,*,*) + 1
      IF(YM .LT. SNOR(KM2(*,*,*))) KM2(*,*,*) = KM2(*,*,*) - 1
C
C      COMPUTE OUTER EDDY VISCOSITY
C
      SNOR(*,1:LEDGE,*) = 0.5*(SNOR(*,1:LEDGE,*) + SNOR(*,2:LEDGE+1,*))
      FFC = FKK*F27*RE*Q1(*,1:LECGE,*)*Q6(*,1:LEDGE,*)
      TMO(L) = FFC*YM*YDUM
      FFCWK = YDUMF*YDUMF*FFC
      UDIFF = ABS(UMAX-UMIN)
      IF(YDUM .GT. UDIFF*YDUMF) TMO  = FFCWK*YM*UDIFF*UDIFF/YDUM
      FIA = FKLEB*SNOR(*,1:LEDGE,*)/YM
      IF(FIA .GT. 1.E5) FIA = 1.E5
      FI = 1.0 + 5.5*FIA**6
      TMO  = TMO /FI
      TMO  = ABS(TMO )
C
C      COMPUTE INNNER EDDY VISCOSITY
C
      TAU = ABS(TAS )
      TMI = Q6*Q1*RE*TAU*(.4*SNOR *(1.-EXP(-RA
     1  *SNOR )))**2
      TMI = ABS(TMI)
C
C      LOAD VISCOSITY COEFFS. INTO ARRAY, USE INNER VALUE UNTIL
C      MATCH POINT IS REACHED
C
      TURMU(*,1:LEDGE,*)=TMI(*,1:LEDGE,*)
      BIT=(TMI(*,1:LEDGE,*).LE.TMO(*,1:LEDGE,*))
      IF(BIT)TURMU(*,1:LEDGE,*)=TMO(*,1:LEDGE,*)
C
C     REARRANGE TURMU(L) SUCH THAT WHEN AVERAGED AT L AND L+1
C        THE CORRECT MIDWAY VALUE WILL BE OBTAINED
C
      SMP = TURMU(*,1,*)
      DO 60 L = 2,LMAX
      TURMS = TURMU*,L,*)
      TURMU*,L,*) = 2.0*SMP-TURMU(*,L-1,*)
   60 SMP = TURMS
      RETURN
      END
```

DIVISION 3

THE THREE-DIMENSIONAL AERODYNAMIC EXPLICIT CODE

DIVISION 3 THE THREE-DIMENSIONAL AERODYNAMIC EXPLICIT CODE

1.0 OVERVIEW

The second metric provided to Control Data analysts was the Hung-MacCormack 'explicit' code which used meshes of 31x31x31 to model three-dimensional corner flows. This metric exhibited several characteristics of interest which differ from the 'implicit' form of solution previously discussed (Division 2). Instead of restructuring the entire code as was done for the Steger-Pulliam code, those routines of greatest interest were essentially dealt with independently, within the code itself. The routines chosen were LX, LYI, and LYC/CHARAC because they constituted the major part of the computation in the explicit code, and because they demonstrate the essentially different characteristics needed for this study.

The LX and LYI subroutines were each vectorized first for the STAR-100 because the programming techniques used are expected to be common for the STAR-100 and the Control Data FMP. Further, this permitted verifying answers against the scalar code, and permitted analysis of the code using the STAR-100 performance counters.

The major points of interest in this section are the findings regarding o short vector lengths (compared to the counterpart vectors in the implicit code);

- 'local' vectorization instead of 'global' vectorization;

- data dependent branching, fundamental to the method of characteristics.

In a later section the effects of these factors on the performance of the FMP will be discussed. What follows is the basic programming considerations employed in vectorizing the key parts of the three-dimensional aerodynamic explicit code.

2.0 OVERALL ANALYSIS

The explicit code is a numerical scheme for solving the three-dimensional Navier-Stokes equations for a supersonic, laminar flow over a compression corner with sidewall effects (ref. 4). There are three directional operators (Lx,Ly,Lz) corresponding to the three coordinate directions. The Ly- and Lz- directional operators are split into suboperators; each of these uses a different method to solve the appropriate equations on the appropriate grid. The three different methods are the explicit method, the Implicit method and the method of characteristics.

The computational procedure for each time step can be outlined in this way:

(1) The Ly- directional operator is executed
    (a) LYC - characteristic method on fine mesh
    (b) LYI - implicit method on fine mesh
    (c) LY - explicit method on coarse mesh
(2) The Lz- directional operator is executed
    (a) LZC - characteristic method on fine mesh
    (b) LZI - implicit method on fine mesh
    (c) LZ - explicit method on coarse mesh
(3) The Lx- directional operator is executed
    (a) LX - explicit method on whole grid
    (b) TURB- turbulence model
    (c) LX - explicit method on whole grid

(4) The Lz- directional operator is again executed
    (a) LZI - implicit method on fine mesh
    (b) LZC - characteristic method on fine mesh
    (c) LZ - explicit method on coarse mesh

(5) The Ly- direction operator is again executed
    (a) LYI - implicit method on fine mesh
    (b) LYC - characteristic method on fine mesh
    (c) LY - explicit method on coarse mesh

To determine the relative effect of vectorization on each of these components, relative timings of the different methods on the 7600 for a 31 x 31 x 31 grid were collected:

Implicit Method

LYI	24.53%
LZI	24.81%
Total	49.34%

Explicit Method

LX	23.58%
LY	5.76%
LZ	6.12%
Total	35.46%

Method of Characteristics

LYC	3.29%	Excluding subroutine CHARAC
LZC	3.33%	Excluding subroutine CHARAC
CHARAC	6.64%	
Total	13.26%	

Miscellaneous other operations

Total 1.94%

Overall Total 100.00%

2.1 Data Handling

The model for the FMP version of the code is to use the same algorithm, but expand the problem to a 100x100x100 grid. The basic variables associated with each grid point are RHO, RHOU, RHOV, RHOW, E, EI, U, V, W, RMUL, giving a basic storage requirement of 10×10^6 data points.

In the proposed FMP the Intermediate Memory contains 32×10^6 words, which is sufficient to hold 1 program of the 10×10^6 size while permitting the staging of the next job to Intermediate Memory during current job execution.

The Main Memory has 4×10^6 words (option to 8×10^6 words), a size not large enough to store the whole problem, thus forcing the algorithm to incorporate some sort of "circular buffering" scheme. The Intermediate Map Unit must be used to read in new data to the Main Memory and write out old data to the Intermediate Memory while the Vector Unit is processing the "current" data.

In addition to the circular buffer, the algorithms must also reorder (or rotate) the data, so that the vectors for the Vector Unit are stored in sequential locations. One can do this in several ways. First, one can do gathers and scatters from/to Intermediate Memory. Second, one can also do a "transposition" within Main Memory or Intermediate Memory. A third option depends upon the groupings of the operators in each time step. One can initially store the data indexed by (k,j,i), then perform the 3 methods of the Ly operator vectorized on "k". The data is stored back or fetched from the Intermediate Memory, in (j,i,k) order for the Lz operator which is vectorized on "j" etc. Stated differently, it is not required that the data be rotated between each of the subroutines. In any case, it is clear that the data handling between the Intermediate and Main Memory is an integral part of the problem.

2.2 Speed Optimization

When launching into a vectorization effort on a new piece of production code, the programmer/analyst is well advised to get reacquainted with the basic tradeoffs inherent in the FMP architecture. The basic execution rate of the Map Units and the degree to which they can be made to operate concurrently with the Vector Unit is a major factor in the vectorization process. In the case of the implicit and explicit codes, it has been determined that the map operations (scatter, gather, transpose) can be almost totally overlapped with other functional unit execution. The 'second order effects' of the FMP architecture then become of great concern.

In particular, the effectivity of the Vector Unit becomes the real measure of the machine performance. The efficiency of operation is tied to the amount of time the unit can be kept productively busy throughout a code execution. The sole contributing factor that limits the unit's full capacity from being utilized is vector startup time. This concept is discussed in the hardware description, however, some elements should be reviewed here. First, startup time is a function of the pipeline configuration for the previous operation, the function now being initiated, and the interdependency of one vector operation with another. Thus a short vector operation whose results must feed another vector operation requires that all the data be stored in memory before the subsequent operation can be started. The time delay required for data to clear the pipelines and be stored back to memory, then brought back from memory to refill the pipelines is, in effect, called startup. Startup time is a fixed overhead assigned to a particular sequence of instructions, and thus the longer the vector to be processed, the less effect that startup time has on overall rates of computation.

Tables I and II are provided to illustrate the effect of this architectural characteristic. These tables represent the <u>relative</u> megaflop rate of the FMP pipelines when compared to operations on vectors of length 30, or when compared to theoretical sequences of vector operations which have zero delay between operations due to the dependency of one vector on another.

Table I

<u>R(N,f,d)</u>	N=	30	100	500	1000	10000
R(30,f,d)	d = 0	1.0	1.66	2.16	2.24	2.324
	d = 5	1.0	2.04	3.16	3.395	3.638
	d = 10	1.0	2.27	4.03	4.46	4.94

Table II

R(N,f,d)	d=	0	5	10
R(N,f,0)	N=30	1.0	.63	.46
	100	1.0	.77	.63
	500	1.0	.93	.87
	1000	1.0	.96	.93

From this example one can conclude that long vectors are much faster. Next, it can be seen that for shorter vector lengths (30-50), reducing the average dependency delay can produce the same increase in speed as going to much longer vectors (500-1000).

The reduction in dependency delay can be accomplished by two techniques. The compiler can schedule a sequence of interdependent vector operations far enough apart, and with other independent operations interspersed, so as to obviate the need for the dependency key being used. This is analogous to the current scalar compilers scheduling a number of interdependent scalar operations by interspersing other operations between the dependent ones, to maximize the use of the floating point bandwidth of the Scalar Unit. The second approach is to maximize the use of long vectors, for the purpose of eliminating the dependency keys. If a vector is sufficiently long, its first elements will be well settled in memory long before the last elements have emerged from the pipelines. These first elements can be 'fetched' from memory for the next vector operation while the current one is still in progress. In this instance the compiler (if it knows that the vectors are long enough at compile time) can eliminate the dependency keys, and the consequent delay.

To achieve long vector operations for the majority of the explicit code execution requires a recoding of the explicit portions such that they no longer compute successive displacements (that is the next value at the current point is based on the just computed new value for the adjacent points), but instead use a process of simultaneous displacements. The metric, as provided, requires the use of successive displacements however, and this fact is reflected in the short vector lengths that were used in its simulation.

3.0 IMPLICIT METHOD

The two routines that use the implicit method are LYI and LZI; these routines account for approximately half the compute time required on the CDC 7600. They are basically the same, except that LYI operates in the Z direction. The LYI routine is discussed below.

The LYI routine operates on "j-pencils" and takes two "half steps" for each pencil. Each half step requires setting up and solving seven tridiagonal systems of equations. The j-pencils are solved successively by a pair of outer DO loops on "i" and "k". The j-pencils only extend over the fine mesh region, which is about half of the total grid.

There are several approaches that may be used to vectorize the LYI routine. One could run the vectors in the "j" direction. This runs head on into the problem of the inherently sequential nature of solving tridiagonal systems by the standard Gaussian elimination scheme; each computation depends on the result of the previous computation. There are several algorithms available for solving tridiagonal systems on vector machines given in reference 5.

Their analysis indicates that the best algorithm is cyclic reduction. However, the timing comparison (albeit for the STAR-100) shows no significant improvement over the standard Gaussian elimination, until the vector lengths are on the order of 250. Since the j-pencil will only have a length of approximately 50 for a 100x100x100 mesh, there is little promise in vectorizing on the "j" direction.

One can run the vectors in the "k" direction. In this case all problems with solving the tridiagonal systems disappear. In addition, the vectors are now of length 100 instead of 50. There is the problem associated with computing the j-pencils simultaneously, rather than successively, as they are in the original scalar code. This means that any terms involving the subscript "k-1" will be using values from the previous time step, rather than the newly computed values from st the (k-1) j-pencil.

One could also run the vectors in the "i" direction; however, the two variables "UP" and "VP" are indexed by "i" masquerading under the names of "K3", "K4", and "K5", and the computations depend heavily on "UP" and "VP", i.e. there is "recursion" on i".

As a final alternative, one could try to vectorize the problem by treating the whole "k" x "i" plane as a vector. This approach would run into the same problem that the "i" vector approach had with the variables "UP" and "VP". In addition, there would be a problem with temporary storage, since the temporary vectors used in the scalar code would now have to be three-dimensional matrices, each of order 100x100x50.

A combination of this last alternative and the second method (running the vectors in the "k" direction) was implemented (see appendix A) and run on the STAR-100 for the ten time steps.

There are differences between the scalar version and the vector version, as expected. The timings per time step on a 31x31x31 grid are shown below.

	7600 Scalar	STAR Scalar	STAR Vector	
LYI	8.2	19.8	4.1	seconds

The recoding of the VLYI subroutine (as the vector version of LYI is called) was kept as straightforward as possible, and every attempt was made to make the statements as similar as possible to the original scalar code. VLYI permits a high degree of local vectorization compared to the implicit code. In this instance the variables to be processed were retained in the same form as in the scalar code, and were transposed within the VLYI subroutine itself, where necessary. This permitted direct replacement of the LYI subroutine with the VLYI subroutine on the STAR-100 to verify the correctness of the final results. To accomplish this a small routine called ROTATE was created for the STAR-100 version. This routine is replaced on the FMP by a vector map operation which performs the rotation while gathering data from Intermediate Memory for the current slab.

In a totally vectorized version of the explicit code these map operations would be moved outside the VLYI subroutine and 'hidden' under the arithmetic operations of routines like TURBDA (which is called before the Y operations are initiated). Local optimization then consisted of further local vectorization, almost on a DO loop by DO loop basis. The rotation process arranged the data such that all arithmetic operations could proceed over vectors of at least KMAX length. Each of the DO loops in J were analyzed to determine if there were recursions in J, and if not, the loop was vectorized for the J direction also, yielding vectors of length KMAX*JMAX/2. For mesh sizes of 100x100x100 this would provide vectors of about 5000 elements, which is considerably shorter than the 60000 elements possible in the implicit code.

In the vectorization of the tridiagonal routine, the maximum vector length can only be KMAX since the solution is recursive in J. The maximum vector length is then 100 in TRIDIAG, which is the same as that in BTRI in the implicit code.

To make the vectorization easier, routines such as GI and DIAG were incorporated in-line, rather than as subroutines. Although it is expected that the compiler will be able to include out-of-line subroutines into in-line object code automatically, the programmer's help is still desirable.

4.0 EXPLICIT METHOD

The three routines that use the explicit method are LX, LY and LZ. These routines are basically the same except for the direction of the operator, and the fact that the LY and LZ routines are restricted to the coarse mesh portion of the grid. The LX routine is discussed below.

The LX routine operates on i-pencils and takes two half steps for each pencil. Each half step solves the explicit equations for the LX operator. The i-pencils are solved successively by a pair of outer DO loops on "k" and "j".

Again there are several ways in which to vectorize the LX routine. The most straightforward way is to run the vectors in the "i" direction. This can be done since the equations are explicit equations. There is some apparent recursion on "i" in the "DO 5 I=2,IE" loop, but a careful examination of the FX subroutine shows that it is, in fact, not recursive on "i". This was implemented (see appendix B) and run for ten time steps on the STAR-100. The timings per time step on a 31x31x31 grid are shown below:

	7600 Scalar	STAR Scalar	STAR Vector	
LX	8.2	16.7	3.6	seconds

Running the vectors in the "j" or "kk" direction will run into the rate of convergence question arising from solving simultaneous pencils, rather than successive pencils. The equations for SIGX, TAUXY, TAUXZ, and DISX show a greater dependence on the "j" index than the "k" index, which implies that it would be preferable to vectorize on "k" rather than "j". This gives the second option of vectorizing the routine on both "i" and "k". This would give a vector length of up to 10,000 rather than 100 and a corresponding speed increase of up to approximately 80% (subject to the dependency delays and size restrictions of Main Memory).

5.0 METHOD OF CHARACTERISTICS

There are two routines LYC and LZC that use the method of characteristics.

The original scalar subroutine CHARAC is computationally very efficient. One can attribute its efficiency to the scheme used to generate the mesh points. The mesh points are generated in such a manner that the solution of the characteristic equations involves simple additions and subtractions. The mesh generation scheme involves some very tricky logic that does not readily lend itself to machines with a pipeline architecture.

Examination of the routine LYC shows that except for the call to CHARAC, the remainder of the routine is easily and directly vectorized in the same local fashion as were LX and LYI. Assuming that all vectorized routines improve uniformly in performance from the 7600 to the FMP, it is necessary then to focus on the potential 'weak spot' in the vectorization--the CHARAC routine itself. This routine, quite obviously, cannot remain scalar in nature if all other routines become highly vectorized, for although it takes up only 6% of the compute time of the 7600, it could be the bottleneck in the FMP performance on the explicit code.

One method of attack in vectorization is given in the routine VCHARAC which is shown in appendix C. The key feature in this approach is the processing of entire planes of data of length IL*KL/2 by both LYC and CHARAC. The problem with this is that in CHARAC each element of the plane may be handled differently from any other element depending upon the data itself.

As in MUTUR (in the implicit code) use of the 'control vector' is introduced; this is a bit-string of ones and zeroes which controls the storage of data, depending on the presence (or absence) of one-bits in the string. In the CHARAC routine the control vector has as many bits in it as the plane (IL*KL/2). Elements in this plane are updated depending upon whether the the corresponding element of the control vector is a one or not. The control vectors may be manipulated and combined using the bit logical operations -- .NOT., .AND., .OR. ... etc.

A brief glance at the listing of CHARAC in appendix C will show that, for the most part, the scalar variable names have been retained from the original CHARAC but have become DYNAMIC variables representing arrays (usually of IL*KL/2 size). This can create some confusion on the part of analysts familiar with the original version and thus some care must be used in reading the listing to remember that almost all operations shown are array operations and not scalar operations.

There are two FORTRAN constructs used here that do not appear in the implicit code: `IF(BITO)Y(*,*,*)=Z(*,*,*)`

and

`Y(*,*)=Z(JLIST(*,*))`

The first construct moves elements from the array Z to the array Y depending upon the presence of a one-bit in the control vector BITO. The second construct subscripts the array Z with an array JLIST. This operates as follows:

The first element of JLIST--JLIST(1,1)--is used as an integer subscript for the array Z. The element at that position in Z is then moved to the first element in the array Y--Y(1,1). The next element of JLIST--JLIST(2,1)--is then used as a subscript of Z and the data moved to Y(2,1). This continues until all elements of JLIST and Y are processed. Obviously JLIST and YLIST must be conformal, but Z need not be. If a subscript in Z is out of range of Z no error message is created.

A third construct used employs the Q8xxxx in-line call for machine language instructions that is referenced but not illustrated in the FORTRAN specification. In the CHARAC routine the call Q8NOBITS(BITO)
determines if there are any one-bits in the string BITO. If there are none, the condition is TRUE and a branch may be triggered by that condition. This call is used to 'bail out' of the DO loops when all elements in the I-K plane have become inactive, to prevent unnecessary passes through the DO loops.

Examining one example of the vectorization technique illustrates how these constructs help in 'parallelizing' CHARAC.

DO loop 10, lines 1730 through 2060 of VCHARAC, shows the means used to process each element in the plane separately, while still performing vector operations.

First, the starting value of the DO loop index could be different for each element in the plane depending on values calculated for elements of JLIST in the preceding loop. Thus the DO 10 loop must be viewed as a parallel set of DO 10 loops (as a matter or fact, a whole plane's worth of DO loops), each with a potentially different starting index. The DO 10 shown is then set up to start at the earliest index found in JLIST (via the function Q8SMIN which returns the scalar minimum of the entire vector JLIST), and the loop 10 could potentially end at JL, unless ended by a 'bail out' earlier.

Once launched into the loop, the immediate need is to find out which elements in the I-K plane to process, first of all, which elements have indexes in JLIST that have not yet reached the limit JL. This is accomplished by the statement at 1870

```
     BIT4=(JLIST.LE.JL)
```

which forms a control vector at vector rates from the conditional test shown. In statement 1880 any elements (bits) from this control vector are eliminated if they have already been 'deactivated' in previous loops. This information is carried in the bit-string BIT5 (and BIT3), and the logical .AND. operation thus provides a BIT4 which represents all active elements at this time.

The 'bail out' check is then made at statement 1890, to skip processing entirely if all elements are inactive.

Statements 1900 through 1930 deal with temporary data areas where there is no need to worry about controlling the data storage. Likewise the updating of YJK2 temporary data need only be controlled by the condition YJK2.GT.Y2. Note that the form shown in lines 1940 and 1950 could be replaced by the original

```
     IF(YJK2.GT.Y2)YJK2=Y2
```

however, the desire was to explicitly show how the hardware actually performs the operation at this point in the listing. In later examples the original scalar form is preserved and the programer must be aware of its vector/control vector nature.

Lines 1960 through 2010 then update key data arrays based on the contents of the control vector. Finally at line 2030, a test is made to determine if other elements should become inactive. For all remaining active elements, the individual indices are then updated in JLIST, and a return is made to the beginning of the loop. It can be seen that elements can become 'deactivated' in this loop by having their individual index reach the limit JL, or by having the computed value of YJK2 for that particular element become equal to the value of Y2.

The computations in lines 1960 through 2010 involve the use of the list of indexes JLIST rather than the scalar DO variable. Thus a potentially different value of Y may be used in the processing for each element of the I-K plane. The operation implied by this involves performing gather operations from the array Y. The Map Unit can perform both gather operations for a single statement such as 1980:

```
     IF(BIT4)VIJK2=WT1*HYV(JLIST)+WT2*HYV(JLIST+1)
```

In this example one map and one vector operation will be generated by the compiler.

The use of index lists and control vectors throughout the other loops in CHARAC follows the same pattern described here. In some cases the DO loop limits rather than the DO indexes themselves are separate entities, and in the case of LOOP 38 both the starting and ending conditions are individualized for every element in the I-K plane.

6.0 IMPLICATIONS

The method of vectorization shown here appeared to be the most straightforward one handy. The use of control vector techniques is not without its penalties, however; as more passes are made through the loops, and more elements are deactivated, the efficiency of this scheme degenerates quickly. NASA and Control Data mathematicians firmly believe that in the domain of real problem solutions, the number of iterations through each loop would be from 3 to 5, as the 'waves' tend to travel closely together in real physical solutions. In this instance then, the unused calculations in the I-K plane would be discarded, but would still take processing time. If a 1/2 reduction in elements is assumed for each pass until the last, then the number of useful operations would be

Pass 1---IL*KL/2
Pass 2---IL*KL/4
Pass 3---IL*KL/8
Pass 4---IL*KL/16
Pass 5---IL*KL/32

while the number of actual operations would be constant for each pass at IL*KL/2 operations. In five passes 5*IL*KL/2 operations would have been done, but only 31*IL*KL/32 operations would have been used in actual results. This can reduce the floating point rate potential of the vectorization by a factor of 31/80.

If it turns out that the number of passes required to resolve all the waves is quite large and the number of residual active elements remaining from pass to pass is quite small, then another approach would be called for. In this instance, the structure of the program would remain the same, but instead of performing controlled storage operations, the choice would be to perform compress and merge operations using the control vector to squeeze the I-K plane down to only the active elements each pass.

In the section on performance evaluation of the metric codes (see Division 1) the impact of this vectorization technique can be seen more clearly.

APPENDIX A

VECTORIZED VLYI ROUTINE

```
*DECK VLYI2
      SUBROUTINE VLYI
C
      COMMON /A11/  RHO(31,31,31) , RHOU(31,31,31) , RHOV(31,31,31)
      COMMON /A12/ RHOW(31,31,31) ,     E(31,31,31) ,    EI(31,31,31)
      COMMON /A13/     U(31,31,31) ,     V(31,31,31) ,     W(31,31,31)
      COMMON /A6 / RMUL(31,31,31)
      COMMON /A3/ Y(31),DYCELL(31),JS1,JE1,JS2,JE2,JLFM,JL,YF,YH,
     1            Z(31),DZCELL(31),KS1,KE1,KS2,KE2,KLFM,KL,ZF,ZH
      COMMON /A4/ ISHK, ILE, IE, IL, K1, K2, K3, K4, K5
      COMMON /A5/ GAMMA, GAMM1, GAMMPR, CV, CV1, STOKES, U0, C0,
     1            P0, RHO0, RL, X0
      COMMON /A7/ DX,DX1,DY,DUMDY1,DZ,DD1,EIWALL,IADBWL,DT,CFL,CONST
      COMMON /A8/ ISMTHX,ISMTHY, ISMTHZ, LYICNT, LYCCNT, LZCCNT, LZICNT,
     1            NLYI,NLZI,BETA,BETA1,CRKNIS
      COMMON /SBC/ SBC(31,31,5) , SBCN
      COMMON /ANGL/ TANT(32), COST(32), TANTH,TANTHB,COSTH,COSTSQ,SECTH
      COMMON /A2/ PDUM(32,5) , CP(32)
      COMMON /TRID/  KLEN
C
      DIMENSION       UP(31,23,3) , VP(31,23,3) ,
     1               UI(31,23)    , VI(31,23)    , WI(31,23)    ,
     2               USQ(31,23)   , VSQ(31,23)   , WSQ(31,23),EII(31,23),
     3               AA(31,23)    , BB(31,23)    , CC(31,23)    ,
     4               AAE(31,23)   , BBE(31,23), CCE(31,23)     ,
     5               FFU(31,23)   , FFV(31,23)   , FFW(31,23)
      DIMENSION FFUSQ(31,23),FFVSQ(31,23),FFWSQ(31,23),FFEI(31,23)
      DIMENSION FUSQ(31,23), FVSQ(31,23), FWSQ(31,23)
      DIMENSION ETA(31,23), RKAPPA(31,23), RLMBDA(31,23)
      DIMENSION F(31,23,5), P(31,31), PRDICT(31,31,5)
      DIMENSION DY1(31,23),DZ1(31,23), DTDRDY(31,23)
      DIMENSION DISX(31,23), DISY(31,23), DISZ(31,23)
      DIMENSION GUPK2(31), GVPK2(31), GWPK2(31)
      DIMENSION RHOI(31,23), RMU(31,23), RK(31,23)
      DIMENSION DYSELL(31,23)
C
      DATA  FORTH /1.3333333333333/
C
      DO 2 J=1,JL
      P(1,J) = CP(J)
    2 CONTINUE
      CALL ROTATE (RHO  , IL, JL, KL)
      CALL ROTATE (RHOU , IL, JL, KL)
      CALL ROTATE (RHOV , IL, JL, KL)
      CALL ROTATE (RHOW , IL, JL, KL)
      CALL ROTATE (E    , IL, JL, KL)
      CALL ROTATE (EI   , IL, JL, KL)
      CALL ROTATE (U    , IL, JL, KL)
      CALL ROTATE (V    , IL, JL, KL)
      CALL ROTATE (W    , IL, JL, KL)
      CALL ROTATE (RMUL , IL, JL, KL)
      TSTR = SECOND (D)
      LYICNT = LYICNT + 1
      JADD       = MOD(LYICNT,2)
      A3         = FORTH
      KLEN = KE2 - 1
      K3 = 1
      K4 = 2
      K5 = 3
C
      DX2 = .5*DX1
C
C - SETUP VECT EQUIV OF DY1,DZ1, AND DYCELL
```

```
      DZ1(1,1)=1.
      DZ1(2,1:KLEN)=1./(Z(3:KLEN)-Z(1:KLEN))
      DO 251 J=1,JE1
      DY1(*,J)=1./(Y(J+1)-Y(J))
      DZ1(*,J)=DZ1(*,1)
      DYSELL(*,J)=DYCELL(J)
251   CONTINUE
C
C- ALL PRIMARY VARIABLES ARE INDEXED BY (K,J,I)
C- VECTORS ARE ON K FOR K=2,KE2  (KLEN=KE2-1)
C - MAIN LOOP IS ON I
C  IN ORDER TO MARCH DOWN STREAM
C
      DO 1 I=2,IE

      TANTH=TANT(I)
      TANTHB=.25*(TANT(I+1)+TANT(I-1)+2.*TANT(I))
      COSTH = COST (I)
      SECTH = 1./COSTH
      COSTSQ = COSTH*COSTH
      TANTM = 1.0 + TANTH*TANTHB
C
C**               **** CALL PRESTY(I,K,1,JS2,0)
C
      PRDICT(*,1:JS2,1) = RHO (*,1:JS2,I)
      PRDICT(*,1:JS2,2) = RHOU(*,1:JS2,I)
      PRDICT(*,1:JS2,3) = RHOV(*,1:JS2,I)
      PRDICT(*,1:JS2,4) = RHOW(*,1:JS2,I)
      PRDICT(*,1:JS2,5) = E   (*,1:JS2,I)
      P(*,1:JS2) = GAMM1*RHO(*,1:JS2,I)*EI(*,1:JS2,I)
      P(*,1)=P(*,2)
C
C
C- BUFFERS FOR UP , VP  K3=I-1  K4=I  K5=I+1
C
      K6=K3
      K3=K4
      K4=K5
      K5=K6

      IF(I.NE.2) GO TO 4
      UP(*,1:JS2,K3)=U(*,1:JS2,1)+TANT(1)*V(*,1:JS2,1)
      VP(*,1:JS2,K3)=V(*,1:JS2,1)-TANT(1)*U(*,1:JS2,1)
      UP(*,1:JS2,K4)=U(*,1:JS2,2)+TANT(2)*V(*,1:JS2,2)
      VP(*,1:JS2,K4)=V(*,1:JS2,2)-TANT(2)*U(*,1:JS2,2)
4     CONTINUE
      UP(*,1:JS2,K5)=U(*,1:JS2,I+1)+TANT(I+1)*V(*,1:JS2,I+1)
      VP(*,1:JS2,K5)=U(*,1:JS2,I+1)-TANT(I+1)*U(*,1:JS2,I+1)
C
C  PASS TWICE FOR TIME-MEAN OF R.H.S.
C
      DO 70 N=1,2
C
C  SET UP IMPLICIT VBLS
C  UI,VI,WI,EII,USQ,VSQ,WSQ
C
      RHOI(*,JS1:JS2)=1./RHO(*,JS1:JS2,I)

      UI(*,JS1:JS2)=(RHOU(*,JS1:JS2,I)+TANTH*
     1    RHOV(*,JS1:JS2,I))*RHOI(*,JS1:JS2)

      VI(*,JS1:JS2)=(RHOV(*,JS1:JS2,I)-TANTH*
     1     RHOU(*,JS1:JS2,I))*RHOI(*,JS1:JS2)
```

```

      WI(*,JS1:JS2)=RHOW(*,JS1:JS2,I)*RHOI(*,JS1:JS2)

      USQ(*,JS1:JS2)=UI(*,JS1:JS2)*UI(*,JS1:JS2)

      VSQ(*,JS1:JS2)=VI(*,JS1:JS2)*VI(*,JS1:JS2)

      WSQ(*,JS1:JS2)=WI(*,JS1:JS2)*WI(*,JS1:JS2)

      EII(*,JS1:JS2)=E(*,JS1:JS2,I)*RHOI(*,JS1:JS2)
     1         -.5*((USQ(*,JS1:JS2)+VSQ(*,JS1:JS2))*
     2            COSTSQ + WSQ(*,JS1:JS2))

8     CONTINUE
      NM1=N-1
      B=1./N
      UI(*,1)=UI(*,2)
      WI(*,1)=WI(*,2)
      VI(*,1)=-VI(*,2)

      USQ(*,1)=USQ(*,2)
      VSQ(*,1)=VSQ(*,2)
      WSQ(*,1)=WSQ(*,2)
      EII(*,1)=EII(*,2)

      IF(I.LT.ILE)GO TO 7

      IF(IADBWL.EQ.0) EII(*,1)=2.*EIWALL-EII(*,2)
      UI(*,1)=-UI(*,2)
      WI(*,1)=-WI(*,2)
7     CONTINUE
      A1 = FORTH
      A2=1.0


C
C************ CALL GI(I,J,K,JJ)
C
      J1=1
      J2=JE1
      J1P=J1+JADD
      J2P=J2+JADD
      K1M=0
      K2M=KE2-1
      K1P=2
      K2P=KE2+1
      RMU(*,1:JE1) = RMUL(*,J1P:J2P,I)
      RK(*,1:JE1) = GAMMA*RMU(*,1:JE1)
      RLMBDA(*,1:JE1) = STOKES*RMU(*,1:JE1)
      F(*,J1:J2,2) = -RMU(*,J1:J2)*(VP(*,J1P:J2P,K2)-
     1     VP(*,J1P:J2P,K3) -TANTH*(UP(*,J1P:J2P,K5)-
     2.    UP(*,J1P:J2P,K3)))*DX2
      F(*,J1:J2,3) = -RLMBDA(*,J1:J2)*SECTSQ*
     1     ((U(*,J1P:J2P,I+1)-U(*,J1P:J2P,I-1))*DX2
     2     +(W(K1P:K2P,J1P:J2P,I)-W(K1M:K2M,J1P:J2P,I))*
     3     DZ1(*,J1:J2))
     4     +2.*RMU(*,J1:J2)*TANTH*(VP(*,J1P:J2P,K5)
     5     -VP(*,J1P:J2P,K3))*DX2
C
C**NOTE - THE INDEX ON THE DELTA W TERM IS OUT OF ARRAY BOUNDS
C        IN THE ABOVE EXPRESSION
C
      F(*,J1:J2,4) = -RMU(*,J1:J2)*(((V(K1P:K2P,J1P:J2P,I)
```

```
     1     -V(K1M:K2M,J1P:J2P,I)-TANTH*(U(K1P:K2P,J1P:J2P,I)
     2     -U(K1M:K2M,J1P:J2P,I)))*DZ1(*,J1:J2))
     3     -(W(*,J1P:J2P,I+1)-W(*,J1P:J2P,I-1))*DX2*TANTHB)
C
C**NOTE - THE INDEX ON DELTA V AND DELTA U IS OUT OF BOUNDS
C
      F(*,J1:J2,5) = TANTH*RK(*,J1:J2)*(EI(*,J1P:J2P,I+1)
     1     -EI(*,J1P:J2P,I-1))*DX2
      FUSQ(*,J1:J2) = (UP(*,J1P:J2P,K4)+UP(*,J1:J2,K4))*F(*,J1:J2,2)
      FVSQ(*,J1:J2) = (VP(*,J1P:J2P,K4)+VP(*,J1:J2,K4))*F(*,J1:J2,3)
      FWSQ(*,J1:J2) = (W(*,J1P:J2P,I)+W(*,J1:J2,I))*F(*,J1:J2,4)
      ETA(*,J1:J2) = TANTM*RMU(*,J1:J2)*DY1(*,J1:J2)
      RKAPPA(*,J1:J2) = TANTM*RK(*,J1:J2)*DY1(*,J1:J2)
C*****  END OF GI SUBR.
C
C
C** NEED TO MODIFY OUTPUT OF GI FOR THE J=1 CASE
C
      FVSQ(*,1) = 0.0
      J1 = JS1
      J2 = JE1
      J1M = J1-1
      J2M = J2-1
      J1P = J1+1
      J2P = J2-1
      DISX(*,J1:J2)= -.5*((UP(*,J1P:J2P,K4)-UP(*,J1:J2,K4))*F(*,J1:J2,2)
     1     +(UP(*,J1:J2,K4)-UP(*,J1M:J2M,K4))*F(*,J1M:J2M,2))
C
      DISY(*,J1:J2)=-.5*((VP(*,J1P:J2P,K4)-VP(*,J1:J2,K4))*F(*,J1:J2,3)
     1     +(VP(*,J1:J2,K4)-VP(*,J1M:J2M,K4))*F(*,J1M:J2M,3))
C
      DISZ(*,J1:J2)=-.5*((W(*,J1P:J2P,I)-W(*,J1:J2,I))*F(*,J1:J2,4)
     1     +(W(*,J1:J2,I)-W(*,J1M:J2M,I))*F(*,J1M:J2M,4))
C
      DTDRDY(*,J1:J2)=(1.0+BETA)*DT/(DYSELL(*,J1:J2)*RHO(*,J1:J2,I))
C
C*****CALL DIAGON
C
      CRKNIS = 1./NLYI
      BB(*,J1:J2)=-DTDRDY(*,J1:J2)*ETA(*,J1:J2)*CRKNIS
      CC(*,J1:J2)=-DTDRDY(*,J1:J2)*ETA(*,J1P:J2P)*CRKNIS
      AA(*,J1:J2)=-(BB(*,J1:J2)+CC(*,J1:J2))
C
      BBE(*,J1:J2)=-DTDRDY(*,J1:J2)*RKAPPA(*,J1:J2)*CRKNIS
      CCE(*,J1:J2)=-DTDRDY(*,J1:J2)*RKAPPA(*,J1P:J2P)*CRKNIS
      AAE(*,J1:J2)=-(BBE(*,J1:J2)+CCE(*,J1:J2))
C
      FFU(*,J1:J2)=UI(*,J1:J2)-DTDRDY(*,J1:J2)*
     1     (F(*,J1:J2,2)-F(*,J1M:J2M,2))
C
      FFV(*,J1:J2)=VI(*,J1:J2)-DTDRDY(*,J1:J2)*
     1     (F(*,J1:J2,3)-F(*,J1M:J2M,3))
C
      FFW(*,J1:J2)=WI(*,J1:J2)-DTDRDY(*,J1:J2)*
     1     (F(*,J1:J2,4)-F(*,J1M:J2M,4))
C
      FFUSQ(*,J1:J2)=USQ(*,J1:J2)-DTDRDY(*,J1:J2)*
     1     (FUSQ(*,J1:J2)-FUSQ(*,J1M:J2M)+2.*DISX(*,J1:J2))
C
      FFVSQ(*,J1:J2)=VSQ(*,J1:J2)-DTDRDY(*,J1:J2)*
     1     (FVSQ(*,J1:J2)-FVSQ(*,J1M:J2M)+2.*DISY(*,J1:J2))
C
      FFWSQ(*,J1:J2)=WSQ(*,J1:J2)-DTDRDY(*,J1:J2)*
```

```
     1      (FWSQ(*,J1:J2)-FWSQ(*,J1M:J2M)+2.*DISZ(*,J1:J2))
C
      FFEI(*,J1:J2)=EII(*,J1:J2)-DTDROY(*,J1:J2)*
     1      (F(*,J1:J2,5)-F(*,J1M:J2M,5)-COSTSQ*
     2      (DISX(*,J1:J2)+DISY(*,J1:J2))-DISZ(*,J1:J2))
C
C**NOTE**  LAST PART OF DIAGON IS SKIPPED
C
C
C**************** END OF DIAGON
C
10    CONTINUE
C
C  MODIFY FOR J=JE1
C
      J=JE1

      FFU(*,J)=FFU(*,J)-CC(*,J)*UI(*,J+1)
      FFV(*,J)=FFV(*,J)-CC(*,J)*VI(*,J+1)*A3
      FFW(*,J)=FFW(*,J)-CC(*,J)*WI(*,J+1)

      FFUSQ(*,J)=FFUSQ(*,J)-CC(*,J)*USQ(*,J+1)
      FFWSQ(*,J)=FFWSQ(*,J)-CC(*,J)*WSQ(*,J+1)
      FFEI(*,J)=FFEI(*,J)-CCE(*,J)*EII(*,J+1)
      FFVSQ(*,J)=FFVSQ(*,J)- FORTH *CC(*,J)*
     1      VSQ(*,J+1)

      CC(*,J)=0.0
      CCE(*,J)=0.0
      IF(NLYI.EQ.1) GO TO 11
C
C  SPECIAL BDRY CONDITION
C

11    CONTINUE
      A1=1.0
      A2=0.0
      IF(I.LT.ILE) A1=-1.0
C
C
      CALL VTRI2(UI,AA,BB,CC,FFU,A1,A2,1.0,JS1,JE1)
      CALL VTRI2(WI,AA,BB,CC,FFW,A1,A2,1.0,JS1,JE1)
      CALL VTRI2(VI,AA,BB,CC,FFV,1.0,A2,A3,JS1,JE1)
      J=JS1-1
      UI(*,J)=-UI(*,J+1)
      VI(*,J)=-VI(*,J+1)
      WI(*,J)=-WI(*,J+1)
      IF(I.GE.ILE) GO TO 12
      WI(*,J)=WI(*,J+1)
      UI(*,J)=UI(*,J+1)
12    CONTINUE
C
C  OLD DO 13 LOOP
C
      J1 = JS1
      J2 = JE1
      J1M = J1-1
      J2M = J2-1
      J1P = J1+1
      J2P = J2+1
      DISX(*,J1:J2)=-.5*((UI(*,J1P:J2P)-UI(*,J1:J2))**2
     1      *CC(*,J1:J2)+(UI(*,J1:J2)-UI(*,J1M:J2M))**2
     2      *BB(*,J1:J2))
```

```
C.
      DISY(*,J1:J2)=-.5*((VI(*,J1P:J2P)-VI(*,J1:J2))**2
     1     *CC(*,J1:J2)+(VI(*,J1:J2)-VI(*,J1M:J2M))**2
     2     *BB(*,J1:J2))*A3
C
      DISZ(*,J1:J2)=-.5*((WI(*,J1P:J2P)-WI(*,J1:J2))**2
     1     *CC(*,J1:J2)+(WI(*,J1:J2)-WI(*,J1M:J2M))**2
     2     *BB(*,J1:J2))
C
      FFUSQ(*,J1:J2)=FFUSQ(*,J1:J2)-2.*DISX(*,J1:J2)
      FFVSQ(*,J1:J2)=FFVSQ(*,J1:J2)-2.*DISY(*,J1:J2)
      FFWSQ(*,J1:J2)=FFWSQ(*,J1:J2)-2.*DISZ(*,J1:J2)
C
      FFEI(*,J1:J2)=FFEI(*,J1:J2)+((DISX(*,J1:J2)+
     1     DISY(*,J1:J2))*COSTSQ+DISZ(*,J1:J2))
13    CONTINUE
      A1=-1.0
      A2=0.0
      CALL VTRI2(USQ,AA,BB,CC,FFUSQ,A1,A2,1.,JS1,JE1)
      CALL VTRI2(WSQ,AA,BB,CC,FFWSQ,A1,A2,1.,JS1,JE1)
      CALL VTRI2 (VSQ,AA,BB,CC,FFVSQ,A1,A2,A3,JS1,JE1)
      IF(IADBWL .EQ. 0 .AND. I .GE. ILE) A1=1.0
      IF(IADBWL .EQ. 0 .AND. I .GE. ILE) A2=2.*EIWALL
      CALL VTRI2 (EII,AAE,BBE,CCE,FFEI,A1,A2,1.,JS1,JE1)
C
C
C  OLD DO 14 LOOP
C
      J1 = JS1
      J2 = JE1
      IF(N .EQ. 2) GO TO 141
C
C N=1 CASE
C
      PRDICT(*,J1:J2,2)=(RHO(*,J1:J2,I)*(UI(*,J1:J2)
     1     -TANTH*VI(*,J1:J2))*COSTSQ+BETA*RHOU(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,3)=(RHO(*,J1:J2,I)*(VI(*,J1:J2)
     1     +TANTH*UI(*,J1:J2))*COSTSQ+BETA*RHOV(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,4)=(RHO(*,J1:J2,I)*WI(*,J1:J2)
     1     +BETA*RHOW(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,5)=(RHO(*,J1:J2,I)*(EII(*,J1:J2)+
     1     .5*((USQ(*,J1:J2)+VSQ(*,J1:J2))*COSTSQ+
     2     VSQ(*,J1:J2)))+BETA*E(*,J1:J2,I))*BETA1
      GO TO 142
141   CONTINUE
C
C N=2 CASE
C
      PRDICT(*,J1:J2,2)=((PRDICT(*,J1:J2,2)+
     1     RHO(*,J1:J2,I)*(UI(*,J1:J2)-TANTH*VI(*,J1:J2))
     2     *COSTSQ)*.5+BETA*RHOU(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,3)=((PRDICT(*,J1:J2,3)+
     1     RHO(*,J1:J2,I)*(VI(*,J1:J2)+TANTH*UI(*,J1:J2))
     2     *COSTSQ)*.5+BETA*RHOV(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,4)=((PRDICT(*,J1:J2,4)+
     1     RHO(*,J1:J2,I)*WI(*,J1:J2))*.5
     2     +BETA*RHOW(*,J1:J2,I))*BETA1
      PRDICT(*,J1:J2,5)=((PRDICT(*,J1:J2,5)+
     1     RHO(*,J1:J2,I)*(EII(*,J1:J2)+.5*((USQ(*,J1:J2)
     2     +VSQ(*,J1:J2))*COSTSQ+WSQ(*,J1:J2))))*.5
     3     +BETA*E(*,J1:J2,I))*BETA1
142   CONTINUE
C
```

```
C  DEFINE K2
C
      K2 = JE1
14    CONTINUE
C
C  SPECIAL B.C. AGAIN
C
      J=JE1
      GUPK2(*)=-(RMU(*,J)*(UI(*,J+1)-
     1     UI(*,J))*DY1(*,J)-F(*,K2,2))*CRKNIS

      GWPK2(*)=-(RMU(*,J)*(WI(*,J+1)-
     1     WI(*,J))*DY1(*,J)-F(*,K2,4))*CRKNIS

      GVPK2(*)=-(A3*RMU(*,J)*(VI(*,J+1)-
     1     VI(*,J))*DY1(*,J)-F(*,K2,3))*CRKNIS

      SBC(*,I,2)=SBC(*,I,2)+(GUPK2(*)
     1     -TANTH*GVPK2(*))*COSTSQ

      SBC(*,I,3)=SBC(*,I,3)+(GVPK2(*)
     1     +TANTH*GUPK2(*))*COSTSQ

      SBC(*,I,4)=SBC(*,I,4)+GWPK2(*)

      SBC(*,I,5)=SBC(*,I,5)+(-RMU(*,J)*
     1     (USQ(*,J+1)-USQ(*,J)+ FORTH *
     2     (VSQ(*,J+1)-VSQ(*,J))*DY1(*,J)
     3     +FUSQ(*,K2)+FVSQ(*,K2))*COSTSQ
     4     -RMU(*,J)*(WSQ(*,J+1)-WSQ(*,J))*DY1(*,J)
     5     +FWSQ(*,K2)-RK(*,J)*(EII(*,J+1)
     6     -EII(*,J))*DY1(*,J)+F(*,K2,5))*CRKNIS

C
C     MODIFY JADD
C
      JADD=MOD (JADD+1,2)
C
C************** CALL PRSETY (I,K,JS1,JE1,2)
C
      RHOI(*,JS1:JE1)=1./PRDICT(*,JS1:JE1,1)

      U(*,JS1:JE1,I)=PRDICT(*,JS1:JE1,2)*RHOI(*,JS1:JE1)

      V(*,JS1:JE1,I)=PRDICT(*,JS1:JE1,3)*RHOI(*,JS1:JE1)

      W(*,JS1:JE1,I)=PRDICT(*,JS1:JE1,4)*RHOI(*,JS1:JE1)

      EI(*,JS1:JE1,I)=PRDICT(*,JS1:JE1,5)*RHOI(*,JS1:JE1)
     1     -.5*(U(*,JS1:JE1,I)**2 +V(*,JS1:JE1,I)**2
     2     +W(*,JS1:JE1,I)**2)

      P(*,JS1:JE1)=GAMM1*PRDICT(*,JS1:JE1,1)*EI(*,JS1:JE1,I)
      IF(JS1.LE.2) P(*,1)=P(*,2)
C
C************ CALL BCY (K,I,I,JS1,JE1)
C
      U(*,1,I)=U(*,2,I)
      W(*,1,I)=W(*,2,I)
      V(*,1,I)=-V(*,2,I)
      EI(*,1,I)=EI(*,2,I)
      IF(I.LT.ILE) GO TO 253
      W(*,1,I)=-W(*,2,I)
```

```
      U(*,1,I)=-U(*,2,I)

      IF(IADBWL.EQ.0) EI(*,1,I)=2*EIWALL-EI(*,2,I)
253   CONTINUE
      IF(I.NE.IE) GO TO 255
      U (*,JS1:JE1,IL)=U (*,JS1:JE1,IE)
      V (*,JS1:JE1,IL)=V (*,JS1:JE1,IE)
      W (*,JS1:JE1,IL)=W (*,JS1:JE1,IE)
      EI (*,JS1:JE1,IL) =EI(*,JS1:JE1,IE)
255   CONTINUE
C
C  SET THE B.C. FOR ENDS OF K-PENCILS
C  *** SCALAR OPERATIONS ***
C
      DO 256 J=JS1,JE1
      U(1,J,I)=U(2,J,I)
      V(1,J,I)=V(2,J,I)
      W(1,J,I)=-W(2,J,I)
      EI(1,J,I)=EI(2,J,I)
      U(KL,J,I)=U(KE2,J,I)
      V(KL,J,I)=V(KE2,J,I)
      W(KL,J,I)=W(KE2,J,I)
      EI(KL,J,I)=EI(KE2,J,I)
      IF (I.LT.ILE) GO TO 256
      U(1,J,I)=-U(2,J,I)
      V(1,J,I)=-V(2,J,I)
      IF(IADBWL.EQ.0) EI(1,J,I)=2.*EIWALL-EI(2,J,I)
256   CONTINUE
C************ END OF BCY
C
C
C   NOW UPDATE UP,VP FOR N=2 PASS
C
      UP(*,1:JS2,K4)=U(*,1:JS2,I)+TANT(I)*V(*,1:JS2,I)

      VP(*,1:JS2,K4)=V(*,1:JS2,I)-TANT(I)*U(*,1:JS2,I)
      IF (I.NE.IE) GO TO 70
      UP(*,2:JE1,K5)=U(*,2:JE1,I+1)+TANT(I+1)*V(*,2:JE1,I+1)

      VP(*,2:JE1,K5)=V(*,2:JE1,I+1)-TANT(I+1)*U(*,2:JE1,I+1)

C
C  - END OF DO 70 N=1,2 LOOP
C
70    CONTINUE
C
C***************** CALL PHSETY(I,K,JS1,JE1,1)
C
      RHO  (*,JS1:JE1,I)=PROICT(*,JS1:JE1,1)

      RHOU(*,JS1:JE1,I)=PROICT(*,JS1:JE1,2)

      RHOV(*,JS1:JE1,I)=PROICT(*,JS1:JE1,3)

      RHOW(*,JS1:JE1,I)=PROICT(*,JS1:JE1,4)

      E   (*,JS1:JE1,I)=PROICT(*,JS1:JE1,5)

C
C - END OF DO 1 I=2,IE LOOP
C
1     CONTINUE
C
```

```
C***********CALL OUTER (JS1,JE1,KS1,KE2)
C
C - DOWNSTREAM B.C. AT I=IL
C
      RHO (KS1:KE1,JS1:JE1,IL) = RHO (KS1:KE2,JS1:JE1,IE)
      RHOU(KS1:KE1,JS1:JE1,IL) = RHOU(KS1:KE2,JS1:JE1,IE)
      RHOV(KS1:KE1,JS1:JE1,IL) = RHOV(KS1:KE2,JS1:JE1,IE)
      RHOW(KS1:KE1,JS1:JE1,IL) = RHOW(KS1:KE2,JS1:JE1,IE)
      E   (KS1:KE1,JS1:JE1,IL) = E   (KS1:KE2,JS1:JE1,IE)

C
C - SKIP B.C. FOR J=JL
C
C - EDGE B.C. AT K=KL
C
C ** SCALAR **
C
      DO 259 J=JS1,JE1
      DO 259 I=2,IE
      RHO (KL,J,I) = RHO (KE2,J,I)
      RHOU(KL,J,I) = RHOU(KE2,J,I)
      RHOV(KL,J,I) = RHOV(KE2,J,I)
      RHOW(KL,J,I) = RHOW(KE2,J,I)
      E   (KL,J,I) = E   (KE2,J,I)
259   CONTINUE

      TONE = SECOND(D)
      TIME = TONE - TSTR
      WRITE (6,900) TIME
  900 FORMAT(1H0,5X,6HTIME =,F10.6,3X,4HSEC.)

      DO 240 J=1,JL
      CP(J) = P(1,J)
  240 CONTINUE
      CALL UNROT (RHO  , IL, JL, KL)
      CALL UNROT (RHOU , IL, JL, KL)
      CALL UNROT (RHOV , IL, JL, KL)
      CALL UNROT (RHOW , IL, JL, KL)
      CALL UNROT (E    , IL, JL, KL)
      CALL UNROT (EI   , IL, JL, KL)
      CALL UNROT (U    , IL, JL, KL)
      CALL UNROT (V    , IL, JL, KL)
      CALL UNROT (W    , IL, JL, KL)
      CALL UNROT (RMUL , IL, JL, KL)
      RETURN
      END
```

APPENDIX B

VECTORIZED VLX ROUTINE

```
      SUBROUTINE VLX
C
C     LX OPERATOR
C
      COMMON/A11/ RHO(31,31,31),RHOU(31,31,31),RHOV(31,31,31)
      COMMON/A12/  RHOW(31,31,31),E(31,31,31),EI(31,31,31)
      COMMON/A13/ U(31,31,31),V(31,31,31),W(31,31,31)
      COMMON/A2/ PRDICT(32,5),P(32)
      COMMON/A3/ Y(31),DYCELL(31),JS1,JE1,JS2,JE2,JLEM,JL,YF,YH
     1         ,Z(31),DZCELL(31),KS1,KE1,KS2,KE2,KLEM,KL,ZF,ZH
      COMMON/A4/ ISHK,ILE,IE,IL,K1,K2,K3,K4,K5
      COMMON/A5/ GAMMA,GAMM1,GAMMPR,CV,CV1,STOKES,U0,C0,P0,RHO0,RL,X0
      COMMON /A6/ RMUL(31,31,31)
      COMMON/A7/ DX,DX1,DY,DY1,DZ, DZ1,EIWALL,IADBWL ,DT,CEL,CONST
      COMMON /A8/ ISMTHX,ISMTHY, ISMTHZ,
     1            LYICNT, LYCCNT, LZCCNT, LZICNT,
     2            NLYI, NLZI, BETA, BETA1, CRKNIS
      COMMON /ANGL/ TANT(32), COST(32), TANTH, TANTHB, COSTH, COSTSQ,
     1            SECTH
      DIMENSION UII(31)
     1            ,RMU(31),RK(31),RLMBDA(31)
     2    ,DYX(31),UYX(31),VYX(31),SIGX(31)
     3    ,TAUXY(31),TAUXZ(31),DISX(31),F(31,5)
     4    ,COEF(31),CII(31),RHOI(31),TMP(31)
      DIMENSION TMP1(31),TMP2(31)
      DATA NNVLX/0/


      TSTRT=SECOND(0)
      NNVLX=NNVLX+1

      DTDX=DT*DX1
      DO 1 K=KS1,KE2
      DO 2 J=JS1,JE2
      M = IL
      PRDICT(1,1;M)=RHO (1,J,K;M)
      PRDICT(1,2;M)=RHOU(1,J,K;M)
      PRDICT(1,3;M)=RHOV(1,J,K;M)
      PRDICT(1,4;M)=RHOW(1,J,K;M)
      PRDICT(1,5;M)=E   (1,J,K;M)
      P(1;M)=GAMM1*RHO (1,J,K;M)*EI(1,J,K;M)
      IF(NNVLX.NE.1)GO TO 400
      IF(K.NE.5)GO TO 400
      IF(J.NE.5)GO TO 400
      WRITE(6,800)(K6,K6=1,5)
      WRITE(6,801)(I,J,K,(PRDICT(I,K6),K6=1,5),I=1,IL)
400   CONTINUE
      DO 4 N=1,2
      IADD=N+1
      NM1=N-1
      B=1./N
C
C************* EQUIVALENT (DO 5 I=1,IE LOOP)
C NOTE  F(K2,-) = F(I,-)
C       F(K1,-) = F(I-1,-)
C***********************************************************************
C COMPUTE  F(I,-)      FOR I=1,IE
C          PRDICT(I,-) FOR I=2,IE
C***********************************************************************
C
C
C- COMPUTE UII
C
```

```
C
C
      M=IE
      UII(1;M)=.5*(U(2,J,K;M)+U(1,J,K;M))

      TMP(1;M)=(3.*U(2,J,K;M)-U(1,J,K;M))*
     1        (3.*U(1,J,K;M)-U(2,J,K;M))

C
C- SCALAR LOOP INSTEAD OF 2 Q8"S
C
      DO 5 I=2,IE
      IF(U(I+I,J,K) .LE. U(I,J,K)) UII(I)=U(I+IADD,J,K)

      IF(TMP(I) .GE. 0.) UII(I)=U(I+IADD,J,K)

5     CONTINUE
C- FIRST POINT
      UII(1)=U(I+IADD,J,K)
C
C************* CALL FX(UII,I,J,K,II)
C              WHERE II=I+IADD
C
      M = IE
      II=I+IADD
      RMU(1;M)=RMUL(II,J,K;M)

      RLMBDA(1;M)=STOKES*RMU(1;M)
      RK(1;M)=GAMMPR*RMU(1;M)

      DY1=1./(Y(J+1)-Y(J-1))
      DZ1=1./(Z(K+1)-Z(K-1))

      DYX(1;M)=.5*(TANT(1;M)+TANT(2;M))*DY1

      UYX(1;M)=U(II,J+1,K;M)-U(II,J-1,K;M)

      VYX(1;M)=V(II,J+1,K;M)-V(II,J-1,K;M)

      SIGX(1;M)=P(II;M)-(RLMBDA(1;M)+2.*RMU(1;M))
     1      *((U(2,J,K;M)-U(1,J,K;M))*DX1-
     2      UYX(1;M)*DYX(1;M))
     3      -RLMBDA(1;M)*(VYX(1;M)*DY1
     4      +(W(II,J,K+1;M)-W(II,J,K-1;M))*DZ1)

      TAUXY(1;M)=-RMU(1;M)*(UYX(1;M)*DY1
     1      +(V 2,J,K;M)-V(1,J,K;M))*DX1
     2      -VYX(1;M)*DYX(1;M))

      TAUXZ(1;M)=-RMU(1;M)*((U(II,J,K+1;M)
     1      -U(II,J,K-1;M))*DZ1+
     2      (W(2,J,K;M)-W(1,J,K;M))*DX1
     3      -(W(II,J+1,K;M)-W(II,J-1,K;M))*DYX(1;M))

      DISX(1;M)=SIGX(1;M)*UII(1;M)
     1      +TAUXY(1;M)*V(II,J,K;M)
     2      +TAUXZ(1;M)*W(II,J,K;M)
     3      -RK(1;M)*((EI(2,J,K;M)-EI(1,J,K;M))*DX1
     4      -(EI(II,J+1,K;M)-EI(II,J-1,K;M))*DYX(1;M))

      F(1,1;M)=PROICT(II,1;M)*UII(1;M)

      F(1,2;M)=PROICT(II,2;M)*UII(1;M)+SIGX(1;M)
```

```
      F(1,3;M)=PRDICT(II,3;M)*UII(1;M)+TAUXY(1;M)

      F(1,4;M)=PROICT(II,4;M)*UII(1;M)+TAUXZ(1;M)

      F(1,5;M)=PROICT(II,5;M)*UII(1;M)+DISX(1;M)

      IF(ISMTHX .EQ. 0) GO TO 25
C
C- SMOOTHING
C
C                     BUMP II SINCE STARTING AT I = 2
      II = II + 1
      M=IL-2
      TMP(2;M)=P(II+1;M)-2.*P(II;M)+P(II-1;M)
      TMP1(2;M)=VABS(P(II+1;M);TMP2(2;M))+2.*VABS(P(II;M);TMP2(2;M))
     1          +VABS(P(II-1;M);TMP2(2;M))

      TMP(2;M)=CONST*TMP(2;M)/TMP1(2;M)
      TMP1(2;M)=GAMMA*GAMM1*VABS(EI(II,J,K;M);TMP2(2;M))
      CII(2;M)=VSQRT(TMP1(2;M);TMP2(2;M))
      COEF(2;M)=TMP(2;M)*(VABS(U(II,J,K;M);TMP2(2;M))+CII(2;M))

      DO 20 K6=1,5
      F(2,K6;M)=F(2,K6;M)-COEF(2;M)*
     1      (PRDICT(3,K6;M)-PRDICT(2,K6;M))
20    CONTINUE
25    CONTINUE
C- END OF FX SUBR.
C
      M=IE-1
      PRDICT(2,1;M)=(NM1*PRDICT(2,1;M)+RHO  (2,J,K;M)
     1      -DTDX*(F(2,1;M)-F(1,1;M)))*B

      PRDICT(2,2;M)=(NM1*PRDICT(2,2;M)+RHOU(2,J,K;M)
     1      -DTDX*(F(2,2;M)-F(1,2;M)))*B

      PRDICT(2,3;M)=(NM1*PRDICT(2,3;M)+RHOV(2,J,K;M)
     1      -DTDX*(F(2,3;M)-F(1,3;M)))*B

      PROICT(2,4;M)=(NM1*PRDICT(2,4;M)+RHOW(2,J,K;M)
     1      -DTDX*(F(2,4;M)-F(1,4;M)))*B

      PROICT(2,5;M)=(NM1*PRDICT(2,5;M)+E    (2,J,K;M)
     1      -DTDX*(F(2,5;M)-F(1,5;M)))*B

C
C************ END OF DO 5 LOOP
C
C
C DECODE  I=2,IE
C
      M=IE-1
      RHOI(2;M)=1./PROICT(2,1;M)

      U   (2,J,K;M)=PRDICT(2,2;M)*RHOI(2;M)

      V   (2,J,K;M)=PRDICT(2,3;M)*RHOI(2;M)

      W   (2,J,K;M)=PRDICT(2,4;M)*RHOI(2;M)

      EI  (2,J,K;M)=PRDICT(2,5;M)*RHOI(2;M)
     1     -.5*(U(2,J,K;M)**2 +V(2,J,K;M)**2
```

```
     2      +W(2,J,K;M)**2 )

      P(2;M)=GAMM1*PRDICT(2,1;M)*EI(2,J,K;M)

C
C  DOWNSTREAM B.C. AT  I=IL
C
C ** SCALAR **
      DO 9  K6=1,5
9     PRDICT(IL,K6)=PRDICT(IE,K6)
C
C************ CALL BCY (K,2,IE,J,J)
C
      M=IE-1
      IF(J .NE. 2) GO TO 30
C
C - BC AT J=2
C
      U(2,1,K;M)=U(2,2,K;M)

      W(2,1,K;M)=W(2,2,K;M)

      V(2,1,K;M)=-V(2,2,K;M)

      EI(2,1,K;M)=EI(2,2,K;M)


C
C  TEST ON I
      IF(ILE .LT. 2) GO TO 30
      M=ILE-1
      W(2,1,K;M)=-W(2,2,K;M)

      U(2,1,K;M)=-U(2,2,K;M)

      IF(IADBWL .EQ. 0) EI(2,1,K;M)=2.*EIWALL-EI(2,2,K;M)
30    CONTINUE
C
C- BC AT J=JL
C
      M=IE-1
      IF(J .LT. JE2) GO TO 35
      U(2,JL,K;M)=U(2,JE2,K;M)

      V(2,JL,K;M)=V(2,JE2,K;M)

      W(2,JL,K;M)=W(2,JE2,K;M)

      EI(2,JL,K;M)=EI(2,JE2,K;M)

35    CONTINUE
C
C  BC AT  I=IL
C
C**SCALAR**
      U(IL,J,K)=U(IE,J,K)

      V(IL,J,K)=V(IE,J,K)

      W(IL,J,K)=W(IE,J,K)

      EI(IL,J,K)=EI(IE,J,K)

C
```

```
C  BC AT   K=2
      IF(K .NE. 2) GO TO 40
      U(2,J,1;M)=U(2,J,2;M)

      V(2,J,1;M)=V(2,J,2;M)

      W(2,J,1;M)=-W(2,J,2;M)

      EI(2,J,1;M)=EI(2,J,2;M)
C
C TEST ON I
      IF(ILE .LT. 2) GO TO 40
      M=ILE-1
      U(2,J,1;M)=-U(2,J,2;M)

      V(2,J,1;M)=-V(2,J,2;M)

      IF(IADBWL .EQ. 0) EI(2,J,1;M)=2.*EIWALL-EI(2,J,2;M)
40    CONTINUE
C
C  BC AT  K=KE2
C
      M=IE-1
      IF(K .NE. KE2) GO TO 50
      U(2,J,KL;M)=U(2,J,KE2;M)

      V(2,J,KL;M)=V(2,J,KE2;M)

      W(2,J,KL;M)=W(2,J,KE2;M)

      EI(2,J,KL;M)=EI(2,J,KE2;M)

50    CONTINUE
C
C-END OF DO 4  N=1,2 LOOP
C
4     CONTINUE
      M=IL-1
      RHO (2,J,K;M)=PRDICT(2,1;M)

      RHOU(2,J,K;M)=PRDICT(2,2;M)

      RHOV(2,J,K;M)=PRDICT(2,3;M)

      RHOW(2,J,K;M)=PRDICT(2,4;M)

      E   (2,J,K;M)=PRDICT(2,5;M)

      IF(NNVLX.NE.1)GO TO 450
      IF(K.NE.5)GO TO 450
      IF(J.NE.5)GO TO 450
      WRITE(6,800)(K6,K6=1,5)
      WRITE(6,801)(I,J,K,(PRDICT(I,K6),K6=1,5),I=1,IL)
450   CONTINUE
800   FORMAT(1H0,3X,1HI,3X,1HJ,3X,1HK,5(3X,9HPRDICT(I, , I2, 1H) )  )
801   FORMAT(1X,3I4,5E15.6)
C
C-END OF DO 2  J=JS1,JE2 LOOP
2     CONTINUE
C
C-END OF DO 1 K=KS1,KS2 LOOP
1     CONTINUE
C
```

```
C************ CALL OUTER (JS1,JE2,KS1,KS2)
C
C
C BC.AT  I=IL
C **SCALAR**
      DO 60 K=KS1,KE2
      DO 60  J=JS1,JE2
      RHO (IL,J,K)=RHO (IE,J,K)

      RHOU(IL,J,K)=RHOU(IE,J,K)

      RHOV(IL,J,K)=RHOV(IE,J,K)

      RHOW(IL,J,K)=RHOW(IE,J,K)

      E   (IL,J,K)=E    (IE,J,K)
60    CONTINUE
C
C BC AT  J=JL
C
      M=IE-1
      DO 70  K=KS1,KE2
      RHO (2,JL,K$M)=RHO (2,JE2,K$M)

      RHOU(2,JL,K$M)=RHOU(2,JE2,K$M)

      RHOV(2,JL,K$M)=RHOV(2,JE2,K$M)

      RHOW(2,JL,K$M) = RHOW(2,JE2,K$M)

      E   (2,JL,K$M)=E    (2,JE2,K$M)
70    CONTINUE
C
C  BC AT  K=KL
C
      DO 80  J=JS1,JE2
      RHO (2,J,KL$M)=RHO (2,J,KE2$M)

      RHOU(2,J,KL$M)=RHOU(2,J,KE2$M)

      RHOV(2,J,KL$M)=RHOV(2,J,KE2$M)

      RHOW(2,J,KL$M)=RHOW(2,J,KE2$M)

      E   (2,J,KL$M)=E    (2,J,KE2$M)

80    CONTINUE

      TONE=SECOND(D)
      TIME=TONE-TSTRT
      WRITE(6,900)TIME
900   FORMAT(1H0,11HVLX TIME =  ,F10.3,4H SEC )

      RETURN
      END
```

APPENDIX C

VECTORIZED VCHARAC ROUTINE

```
      SUBROUTINE VCHARAC(Y,DYCELL,JLFM,JL,COSTH)
      DYNAMIC HYC,HYP,HYV,HYRHO,HYP
      DYNAMIC NDT,DTC,DYC,YJC,YJCB,YJCT,JCLN,SUM1,SUM3
      DYNAMIC WT1,WT2,RI,PI,VI,CI,DYI,JCADD,Y1,YJK2,VIJK2,PIJK2,CIJ1,Y2
      DYNAMIC CIJ2,VIJK1,PIJK1,JCLNP1,DPDYI,DVDYI,SUM2,SUM4,YJCET,YJCSB
      DYNAMIC YCELLB,YCELLT,VT,PT
      DEFINE (HYC,PRDICT(1:IL,1:KL,1,4)),(HYRHO,PRDICT(1:IL,1:KL,1,1))
      DEFINE (HYP,PRDICT(1:IL,1:KL,1,2)),(HYV,PRDICT(1:IL,1:KL,1,3))
C

C    CHARAC PROCESSES ALL DATA AS PLANES OF IL BY KL ELEMENTS,EACH
C    PLANE AT A DIFFERENT VALUE OF J.
C
      LMT=300
      YF=Y(*,*,JLFM)+.5*DYCELL(*,*,JLFM)*(1.+DYCELL(*,*,JLFM)/
     1    DYCELL(*,*,JLFM-1)
      DT1=1./DT
      COSTSQ=COSTH**2
C    *** FROM SUBROUTINE JCLMN
      JLM=0
      JLN=JLN1
      DYN=DYCELL(2)
      YL=YL1
      DO 51 N=1,2
      NDT=2.*DT*HYC(*,*,2,1)/DYN+1
      CDTC=2.0*DT/NDT
      DYC=DTC*HYC(*,*,2,1)
      YJC=-0.5*DYC
      DEFINE (YJCB,(1:IL,1:KL)),(YJCT,(1:IL,1:KL))
      YJCB=0.0
      YJCT=0.0
      J=1
      JCL=LMT
      DEFINE (JCLN,(1:IL,1:KL)),(SUM1,(1:IL,1:KL)),(SUM3,(1:IL,1:KL))
      JCLN=JCL
C    THIS BROADCASTS THE CONSTANT JCL THROUGHOUT A MATRIX IL BY KL ELEMEN
      SUM1=0.0
      SUM3=0.0
      DEFINE (JLIST,(I:IL,K:KL))
      DO 5 JC=2,JCL
      BIT0=(YJCB.LT.YL.AND.YJCT.GT.YL)
C
C    THIS STATEMENT GENERATES ONE VECTOR OPERATION WHICH CREATES
C    TWO BIT STREAMS REPRSENTING THE CONDITIONS BEING TESTED FOR
C    THE ENTIRE I-K PLANE AT J=1
C      THE BITS STRINGS ARE THEN 'ANDED' TOGETHER BY THE SCALAR UNIT
C      AT THE RATE OF 64 BITS EVERY FOUR CLOCK CYCLES,AND THE
C    CONSEQUENT BRANCH INSTRUCTION EXECUTED.
C
      IF(Q8NOBITS(BIT0))GO TO 6
C
C      WHERE Q8NOBITS IS A MACHINE LANGUAGE CALL TO THE DATA FLAG
C     BRANCH TEST.ANY FUNCTION BEGINNING WITH THE SYMBOLS 'Q8'
C      IDENTIFIES AN IN LINE MACHINE LANGUAGE INSTRUCTION,WHICH
C       CANNOT BE INVOKED BY NORMAL FORTRAN
C
      IF(BIT0)JCLN=JC
*     IF(BIT0)YJC=YJC+DYC
C
C    THIS CONSTRUCT GENERATES THE MACHINE LANGUAGE CONTROL VECTOR
C     OPERATION WHEREIN THE ARITHMETIC STATEMENT IS EXECUTED
C     FOR EVERY ELEMENT OF THE I-K PLANE WHERE THE CORRESPONDING
C     BIT IN THE STRING BIT0 IS A ONE.
```

```
C
      DEFINE (JLIST,(1:IL,1:KL))
      JLIST=0
C
C     JLIST WILL BE USED AS AN INDEX VECTOR DEPENDING ON THE VALUES
C     GENERATED BY THE FOLLOWING:
C
      DO 3 JJ=1,JL
      BIT1=(YJC.GT.Y(*,*,J).AND.YJC.LE.Y(*,*,J+1))
      IF(Q8NOBITS(BIT1)GOTO 4
      J=J+1
      BIT1=BIT1.AND.BIT0
C
C    WE ARE ONLY GOING TO PROCESS ELEMENTS IN THE PLANE THAT ARE
C        STILL ACTIVE
C
      IF(BIT1)JLIST=JLIST+1
C
C     WHEREVER THE CONDITION IS MET,JLIST WILL BE UPDATED
C
 3    CONTINUE
 4    CONTINUE
C
C     WE NOW HAVE A LIST OF INDEXES FOR EVERY I-K ELEMENT THAT IS
C        ACTIVE.THIS LIST CAN BE USED AS A MEANS TO GATHER
C        THE CORRESPONDING ELEMENTS FROM VARIOUS FLOATING POINT ARRAYS
C
      WT1=(Y(JLIST+1)-YJC)/(Y(JLIST+1)-Y(JLIST))
C
C     THIS OPERATION GENERATES ONE GATHER OPERATION USING JLIST INDICES
C        INTO THE ARRAY Y.
C
      WT2=1.0-WT1
      RI(JC)=WT1*HYRHO(JLIST)+WT2*HYRHO(JLIST+1)
      PI(JC)=WT1*HYP(JLIST)+WT2*HYP(JLIST+1)
      VI(JC)=WT1*HYV(JLIST)+WT2*HYV(JLIST+1)
      CI(JC)=WT1*HYC(JLIST)+WT2*HYC(JLIST+1)
      DYC=DTC*CI(JC)
C
C     REMEMBER AT THIS POINT THAT RI,PI,VI,AND CI CONSIST OF
C        PLANES OF DATA IL BY KL IN SIZE,WITH A PLANE FOR
C        EVERY ACTIVE VALUE OF JC
C
      YJCB=YJCT
      YJCT=YJC+0.5*DYC
      DYI(JC)=YJCT-YJCB
      BIT2=(JC.LE.NDT+1)
      IF(BIT2)SUM1=SUM1+PI(JC)
      IF(BIT2)SUM3=SUM3-VI(JC)
 5    CONTINUE
 6    CONTINUE
      DEFINE (JCADD,(1:IL,1:KL))
      JCADD=0
      BIT3=(NDT+3-2*JCLN.GT.0)
      IF(BIT3)JCADD=(NDT+3-2*JCLN)
      JCMAX=JCLN+NDT-JCADD
C
C     WRITE STATEMENT ELIMINATED HERE TEMPORARILY
C
      IF(Q8NOBITS(.NOT.BIT2))GOTO 12
C
C        BIT2 INDICATED WHERE THERE WERE VALUES GREATER THAN ZERO IN
C     JCADD,IF THE ENTIRE PLANE OF JCADD IS ZERO SKIP THE
```

```
C     NEXT PART COMPLETELY
C
IF(BIT2)Y1=YJCT
      BIT3=(Y1.GT.Y(JLIST+1))
      BIT3=BIT3.AND.BIT2
C
C     WE ARE ONLY INTERESTED IN ACTIVE ELEMENTS IN THE PLANE AT THIS TIM
C
      IF(BIT3)JLIST=JLIST+1
C
C     ANOTHER CONTROL VECTOR UPDATE OF THE INDEXED LIST
C
      IF(BIT3)WT1=(Y(JLIST+1)-Y1)/(Y(JLIST+1)-Y(JLIST))
C
C     THIS STATEMENT REQUIRES TWO NEW GATHER OPERATIONS FROM THE
C     ARRAY Y WHICH IS HELD IN MAIN MEMORY....
C
      IF(BIT3)WT2=1.0-WT1
      IF(BIT3)YJK2=Y1
      IF(BIT3)VIJK2=WT1*HYV(JLIST)+WT2*HYV(JLIST+1)
      IF(BIT3)PIJK2=WT1*HYP(JLIST)+WT2*HYP(JLIST+1)
      IF(BIT3)CIJ1=WT1*HYC(JLIST)+WT2*HYC(JLIST+1)
      J1LIST=JLIST
      Y2=Y1+JCADD*DTC*CIJ1
      DO 9 K=1,2
      JLIST=J1LIST
      DO 7 JJ=1,JL
      BIT4=(Y2.GT.Y(*,*,JLIST).AND.Y2.LE.Y(*,*,JLIST+1))
      BIT4=.NOT.BIT4
      IF(BIT4)JLIST=JLIST+1
 7    CONTINUE
      IF(BIT3)WT1=(Y(JLIST+1)-Y2)/(Y(JLIST+1)-Y(JLIST))
      IF(BIT3)WT2=1.0-WT1
      IF(BIT3)CIJ2=WT1*HYC(JLIST)+WT2*HYC(JLIST+1)
      IF(BIT3)Y2=Y1+JCADD*DTC*.5*(CIJ1+CIJ2)
 9    CONTINUE
C
C     THE DO 10 LOOP MUST BE REPLACED COMPLETELY,SINCE EACH ELEMENT
C     IN THE I-K PLANE CAN BE ADVANCED INDEPENDENTLY FROM ANOTHER
C     ELEMENT  N THE PLANE.SINE IN THEORY THE STARTING INDEX J1
C WILL BE DIFFERENT FOR EACH.IN ADDITION ALL OF THE OPERATIONS
C     ARE UNDER THE FURTHER CONTROL OF BIT3 WHICH INDICATES
C     WHICH ELEMENTS IN THE PLANE ARE ACTIVE.ELEMENTS ARE FURTHER
C     DEACTIVATED BY THE STATEMENT IF(YJK2.EQ.Y2)GO TO 11 AT THE
C     BOTTOM OF THE ORIGINAL SCALAR LOOP
C     SO WE MUST GENERATE A NEW CONTROL VECTOR WHICH WILL CONTROL
C     OPERATIONS IN THE REMAINDER OF THIS SEQUENCE!
C
      JLIST=J1LIST
      BIT5=BIT3
      DO 10 JJ=1,JL
      BIT4=(JLIST.LE.JL)
      BIT4=BIT4.AND.BIT5)
      IF(Q8NOBITS(BIT4)GOTO 11
      YJK1=YJK2
      VIJK1=VIJK2
      PIJK1=PIJK2
      YJK2=Y(JLIST+1)
      BIT6=(YJK2.GT.Y2)
      IF(BIT6)YJK2=Y2
      IF(BIT4)WT1=(Y(JLIST+1)-YJK2)/(Y(JLIST+1)-Y(JLIST))
      IF(BIT4)WT2=1.0-WT1
      IF(BIT4)VIJK2=WT1*HYV(JLIST)+WT2*HYV(JLIST+1)
```

```
      IF(BIT4)PIJK2=WT1*HYP(JLIST)+WT2*HYP(JLIST+1)
      IF(BIT4)SUM1=SUM1+JCADD*0.5*(PIJK1+PIJK2)*(YJK2-YJK1)/(Y2-Y1)
      IF(BIT4)SUM3=SUM3-JCADD*0.5*(VIJK1+VIJK2)*(YJK2-YJK1)/(Y2-Y1)
      IF(BIT4)JLAST=JLIST
      BIT6=(YJK2.EQ.Y2)
      BIT5=BIT5.AND..NOT.BIT6
      IF(BIT4.AND.BIT5)JLIST=JLIST+1
 10   CONTINUE
      JLIST=JLAST
      IF(BIT4)CIJ2=WT1*HYC(JLIST)+WT2*HYC(JLIST+1)
      IF(BIT4)DYC=DTC*CIJ2
      IF(BIT4)YJC=Y2-0.5*DYC
 12   CONTINUE
      JCLNP1=JCLN+1
C
C     THE DO 20 LOOP MUST ALSO BE RECONSTRUCTED SINCE THE STARTING
C     VALUE FOR JC COULD BE DIFFERENT FOR EVERY ELEMENT IN THE I-K PLANE
C
      JCLIST=JCLNP1
      JCSTART=Q8MIN(JCLIST)
C
C     THE Q8 FUNCTION HERE RETURNS THE SCALAR MINIMUM OF THE ENTIRE
C     VECTOR JCLIST
C
      DO 20 JC=JCSTART,LMT
      BIT6=(JCLIST+JCADD.GT.JCLN+NDT)
      BIT6=.NOT.BIT6
      IF(Q8NOBITS)GOTO 21
      IF(BIT6)YJC=YJC+DYC
      DO 17 JJ=1,JL
      BIT7=(YJC.GT.Y(JLIST).AND.YJC.LE.Y(JLIST+1))
      BIT7=.NOT.BIT7
      IF(Q8NOBITS(BIT7))GOTO 18
      IF(BIT6.AND.BIT7)JLIST=JLIST+1
 17   CONTINUE
 18   CONTINUE
      IF(BIT6)WT1=(Y(JLIST+1)-YJC)/(Y(JLIST+1)-Y(JLIST))
      IF(BIT6)WT2=1.0-WT1
      IF(BIT6)PI(JCLIST)=WT1*HYP(JLIST)+WT2*HYP(JLIST+1)
      IF(BIT6)VI(JCLIST)=WT1*HYV(JLIST)+WT2*HYC(JLIST+1)
C
C     NOTE THAT THE PRECEEDING TWO STATEMENTS WILL RESULT IN
C     BOTH GATHER AND SCATTER OPERATIONS UNDER THE CONTROL OF
C     BIT VECTOR BIT6.
C
      IF(BIT6)DYC=(WT1*HYC(JLIST)+WT2*HYC(JLIST+1)*DTC
      BIT7=(JCLIST+JCADD.GT.NDT+1)
      IF(BIT6.AND..NOT.BIT7)SUM1=SUM1+PI(JCLIST)
      IF(BIT6.AND..NOT.BIT7)SUM3=SUM3-VI(JCLIST)
      IF(BIT6)JCLIST=JCLIST+1
 20   CONTINUE
 21   CONTINUE
      SUM2=SUM1
      SUM4=SUM3
      YJCT=0.0
C
C     NOW IN DO 30 ALL ELEMENTS IN THE PLANE ARE STARTED TOGETHER
C     HOWEVER THE TERMINATING CONDITIONS COULD BE DIFFERENT
C     FOR EACH ELEMENT DEPENDING ON THE CORRESPONDING VALUES IN THE
C     ARRAY JCLN
C
      JCEND=Q8MAX(JCLN)
C
```

```
C     WE MUST SET THE END OF THE LOOP TO THE MAXIMUM VALUE IN JCLN
C
      DO 30 JC=2,JCEND
      BIT5=(JC.LE.JCLN)
      JCM=JC-NDT+JCADD
      DEFINE (SNG,(1:IL ,1:KL))
      SNG=1.0
      BIT4=(JCM.LE.1)
      IF(BIT4)SNG=-1.
      IF(BIT4)JCM=3-JCM
      JCP=JC+NDT-JCADD
      IF(BIT5)SUM1=SUM1+PI(JC)
      IF(BIT5)SUM2=SUM2+PI(JCP)
C
C     NOTE THAT THE USE OF JCP AS AN INDEX IMPLIES A GATHER OPERATION,
C     WHILE THE USE OF JC AS AN INDEX IMPLIES A SCALAR BROADCAST
C
      IF(BIT5)SUM3=SUM3+VI(JC)
      IF(BIT5)SUM4=SUM4-VI(JCP)
      IF(BIT5)
      1   DPDYI(JC)=-((SUM1-SUM2)/(COSTSQ*RI(JC)*CI(JC))+SUM3-SUM4)/
      2             (2*(NDT+1))-VI(JC))*RI(JC)*(DT1*COSTSQ)
      IF(BIT5)
      1   DVDYI(JC)=-((SUM1+SUM2*(RI(JC)*CI(JC)*COSTSQ)*(SUM3+SUM4))/
      2             (2*(NDT+1))-PI(JC))/(DT*GAMMA*PI(JC))
      IF(BIT5)YJCB=YJCT
      IF(BIT5)YJCT=YJCB+DYI(JC)
      BIT6=(JC.EQ.JCLN)
      IF(.NOT.BIT6)SUM1=SUM1-PI(JCM)
      IF(.NOT.BIT6)SUM2=SUM2-PI(JC)
      IF(.NOT.BIT6)SUM3=SUM3-SNG*VI(JCM)
      IF(.NOT.BIT6)SUM4=SUM4+VI(JC)
C
C     AGAIN SUM1 AND SUM3 COMPUTATION HERE REQUIRE GATHER OPERATIONS
C     USING THE LIST JCM,WHILE SUM2 AND SUM4 ARE SCALAR BROADCASTS
 30   CONTINUE
 31   CONTINUE
      IF(JLN.NE.JLFM)GOTO 32
      JCLIST=JCLN
      PT(JLFM+1)=(SUM1+SUM2+(RI(JCLIST)*CI(JCLIST)*COSTSQ)*(SUM3+SUM4))/
      1      (2*(NDT+1))+(YP-0.5*(YJCB+YJCT))*DPDYI(JCLIST)
 32   CONTINUE
      DEFINE (YCELLT,(1:IL,1:KL)),(YJCSB,(1:IL,1:KL))
      DEFINE (JCE,(1:IL,1:KL))
      VT(*,*,1)=0.
      PT(*,*,1)=P0
      DO 40 J=2,JLN
      YJCET=YJCSB
      YCELLB=YCELLT
      YCELLT=YCELLB+DYCELL(*,*,J)
      JCS=JCE
      JCSTART=Q8MIN(JCS)
      DEFINE (BIT8,(1:IL,1:KL))
      BIT8=.TRUE.
      DO 35 JC=JCSTART,LMT
      IF(BIT8)JCE=JC
      IF(BIT8)YJCET=YJCET+DYI(JC)
      BIT4=(YJCET.GT.YCELLT)
      BIT8=BIT8.AND..NOT.BIT4
      IF(Q8NOBITS(BIT8)GOTO 36
 35   CONTINUE
 36   CONTINUE
      DEFINE (DYDY,(1:IL,1:KL)),(DPDY,(1:IL,1:KL))
```

```
      DVDY=0.0
      DPDY=0.0
      YJCT=YJCSB
C
C     AS IN THE PREVIOUS LOOP 20 AND LOOP 30 CASES LOOP 38 WILL
C     BEGIN AND END AT DIFFERENT POINTS FOR EACH ELEMENT OF
C     THE I-K PLANE
C
      JCSTART=Q8MIN(JCS)
      JCEND=Q8MAX(JCE)
      JCLIST=JCS
      DEFINE (WT,(1*IL,1!KL))
      DO 38 JC=JCSTART,JCEND
      BIT9=(JCLIST.LE.JCE)
      IF(BIT9)YJCB=YJCT
      IF(BIT9)DYC=DYI(JCLIST)
      IF(BIT9)YJCT=YJCB+DYC
      IF(BIT9)WT=1.0
      IF(YJCB.LT.YCELLB)WT=WT-(YCELLB-YJCB)/DYC
      IF(YJCT.GT.YCELLT)WT=WT-(YJCT-YCELLT)/DYC
C
C     ALTHOUGH THESE LAST TWO STATEMENTS APPEAR TO BE SCALAR
C     THE IF TEST AND ARITHMETIC ARE OVER THF ENTIRE I-K PLANE
C
      IF(BIT9)DVDY=DVDY+WT*DVDYI(JCLIST)*DYC/DYCELL(J)
      IF(BIT9)DPDY=DPDY+WT*DPDYI(JCLIST)*DYC/DYCELL(J)
 38   CONTINUE
      YJCSB=YJCET-DYC
      IF(J.LE.JLM)GO TO 40
      VT(*,*,J)=VT(*,*,J-1)+DVDY*DYCELL(J)
      PT(*,*,J)=PT(*,*,J-1)+DPDY*DYCELL(J)
 40   CONTINUE
 50   CONTINUE
      IF (JLN.E.JLFM)GOTO 52
      DYN=DYN1
      JLM=JLN
      JLN=JLFM
      YL=YF
 51   CONTINUE
 52   CONTINUE
      RETURN
      END
```

DIVISION 4

WEATHER/CLIMATE

APPLICATION STUDY

DIVISION 4

WEATHER/CLIMATE APPLICATION STUDY

1.0 INTRODUCTION

A portion of this study effort was to be devoted to analyzing operation of the FMP on codes other than aerodynamic flow simulations. For this purpose, the application of weather/ climate modeling was selected and NASA provided two codes to be studied. One code was developed by Goddard Institute for Space Studies (GISS); the other code was developed at Massachusetts Institute of Technology (MIT). Sections 2 and 4 present some specific aspects and salient points of the GISS model and MIT model, respectively. Section 3 briefly discusses investigation of portions of the GISS code, and section 5 provides analysis of the MIT (spectral) code. Some background and significant events leading to these two models was supplied to NASA-Ames as a separate, incidental report.

These two models represent, to some extent, current state-of-the-art in dynamical forecasting with numerical methods. They differ from each other, among other things, mainly in their approaches: the GISS model employs the finite difference method whereas the MIT model employs the spectral method. Both are more or less designed for the purpose of long term prediction rather than the day-to-day weather forecasting. However, there is no fundamental reason why they could not be extended to predict daily weather. In the MIT model, the quasi-geostrophic approximation is imposed to conserve computations. For the operation of day-to-day global prediction, the quasi-geostrophic approximation is not very desirable and therefore has to be relaxed. On the other hand, the Australian meteorological community has successfully put the spectral method, without the quasi-geostrophic approximation, into operation for predicting weather.

Although both are more for climate than weather, the GISS model is mainly concerned with the dynamical development in the troposphere, the climate/weather in which man lives and that which is commonly discussed. The MIT model, on the other hand, is geared to the state of the upper atmosphere which certainly affects the troposphere below. Consequently, in the GISS model the effect of radiation, such as ozone absorption, is parameterized in some tractable manner for the purpose of calculating the circulation. In the MIT model, the production and destruction, as well as transportation, of ozone are computed explicitly using the dynamical model as a vehicle. Table 1 tabulates some of their differences for a closer comparison.

Symbols used in this report are listed and defined in table 2.

TABLE 1. Characteristics of the GISS and MIT Models

Characteristic	GISS Model	MIT Model
Method	Finite difference	Spectral
Prediction time scale	Medium (wks/mos)	Very long (yrs)
Dynamic system	Primitive	Quasi-geostrophic
Time step (Δt)	5 minutes	1 hour
Vertical coordinate	σ	ln P
Number of layers		
Total	9	25
In troposphere	8	6
In stratosphere	1	19
Pressure at top	10 mb	0.04 mb
Height at top	30 km	72 km
Horizontal resolution		
Grid ($^\circ$lat x $^\circ$long)	4 x 5	~15 x 15
Waves	~20	6

TABLE 2. Symbol Definitions

a	radius of earth
C	rate of condensation
c_p	specific heat at constant pressure
E	rate of evaporation
F	horizontal frictional force
f	Coriolis parameter ($=2\omega\mu$)
G	production/destruction rate of ozone
G′	deviation from horizontal average

H_0	scale height
$h\nu$	an ultraviolet photon
$h(Z)$	an empirical infrared cooling coefficient
I	solar flux incident on the top of the atmosphere
J	meridional wave number resolution
k	vertical unit vector
K_d	vertical diffusion coefficient
k_0, k_1, k_3, k_4, k_5	parameters of chemical reaction
M	planetary wave number resolution
N	number of ozone molecules in the column
P	pressure ÷ 1000 mb
$P_{m,n}$	associated Legendre polynomial
p	pressure
p_t	pressure at top of model atmosphere, constant
p_s	pressure at bottom of model atmosphere
Q	heating rate per unit mass
q	water vapor mixing ratio or specific humidity
R	gas constant
S	stability
T	temperature
T^*	an "equilibrium" temperature
t	time
T', ϕ'	deviation of temperature and geopotential from the standard atmosphere distributions
V	horizontal velocity
W	dZ/dt
χ_j	number of mixing ratio of specie j ($j = O_3$, OH, HO_2, NO_2)
χ_{O_3}	ozone number mixing ratio
$\overline{\chi}_{O_3}$	horizontal average
χ'_{O_3}	deviation from horizontal average

$Y_{m,n}$	spherical harmonics of order m, degree n
Z	vertical coordinate ($= -\ln P$) of the MIT model
α	absorption coefficient
β	mass of an average molecule
γ	catalyst
ϵ	energy of a photon of wavelength Λ
ζ	solar zenith angle
η	number density of the "neutral" atmosphere
θ	potential temperature
Λ	optical wavelength
λ	longitude (counted eastward from Greenwich)
μ	sine of latitude
ξ	vorticity
π	$p_s - p_t$
ρ	density
σ	vertical coordinate $\left\{= (p-p_t)/(p_s-p_t)\right\}$ of the GISS model
τ	index for simulated time
Φ	geopotential
ψ	stream function for horizontal velocity
$\psi_m, \psi_{m,n}$	Fourier and spectral coefficients of ψ
$\psi^*_m, \psi^*_{m,n}$	complex conjugate
ω	angular velocity of the earth
$\partial X/\partial P$	velocity potential for horizontal velocity
∇	spherical gradient operator

2.0 THE GISS MODEL

The GISS model is a 9-layer primitive equation model. Its evolution began with models developed at UCLA by Arakawa and Mintz, in particular their 3-level model. Consequently, the GISS model shares the overall structure of the UCLA model and retains the vertical coordinate formulation, the Arakawa scheme with advective quasi-conservation of important quadratic quantities, as well as much of the UCLA representation of physical processes occurring at or near the lower boundary of the atmosphere.

It differs from the UCLA model, however, not only in having finer vertical resolution, but also in its treatment of four crucial areas of physical processes, namely, moist convection, turbulent subgrid-scale processes, solar radiation, and long-wave (terrestrial) radiation. As the GISS model was used for observing system simulation experiments, asynoptic data assimilation studies, and experimental long-range forecasting, it was not bound by the very high horizontal resolution nor the stringent real-time requirements of the operational numerical weather prediction. Also, those aspects which are of importance only to very long time-scale are simplified because the GISS model is intended for medium-range forecast. The model was verified by the short-range forecast, and by integration which is intermediate in length between the few days of the operational prediction and the seasonal, annual, or multi-annual period of climate simulation.

It includes a realistic distribution of continents, oceans, and topography. Detailed calculations of energy transfer by solar and terrestrial radiation make use of cloud and water vapor fields calculated by the model. The hydrological cycle of the model includes two precipitation mechanisms: large-scale supersaturation and a parameterization of subgrid-scale cumulus convection.

Numerical integration requires about 70 minutes of IBM 360/95 computer time for each simulated day. In the interest of computational economy, most aspects of the model representations of physical processes (other than advection) are calculated only for every half-hour of simulated time. These processes include surface interaction and hydrology. Similarly, solar and terrestrial radiation calculations are performed, however, only for every 2 hours of simulated time.

The system of governing equations for the GISS model is displayed in figure 1. It consists of the following:

(1) the equation of motion,
(2) the equation of continuity,
(3) the equation of state,
(4) the first law of thermodynamics,
(5) the hydrostatic approximation,
(6) the conservation equation of water vapor.

The behavior of the atmosphere is represented primarily by the following dependent variables:

- the temperature,
- the specific humidity (mixing ratio) of water vapor,
- the difference of surface pressure and pressure at model top,

- the horizontal velocity with the components zonal wind and meridional wind.

The basic independent variables are:

- time,
- latitude,
- longitude,
- the vertical coordinate.

The atmostphere is modeled to have a vertical resolution of 9 evenly divided layers. The top of the atmosphere is taken as a 10 mb isobaric surface. It should be noted, however, that in the treatment of terrestrial radiation, the number of layers is doubled and in addition, 2 more layers are introduced to extend the model atmosphere from 10 mb to 1 mb, in order to obtain better estimates of the radiative fluxes.

$$\frac{\partial}{\partial t}V = -\left\{(V\cdot\nabla)V + f\,k \times V + \nabla\Phi + \frac{\sigma}{\rho}\nabla\pi + \dot{\sigma}\frac{\partial}{\partial\sigma}V\right\} + F \tag{1}$$

$$\frac{\partial}{\partial t}\pi = -\left\{\nabla\cdot(\pi V) + \frac{\partial}{\partial\sigma}(\pi\dot{\sigma})\right\} \tag{2}$$

$$\frac{p}{\rho} = RT \tag{3}$$

$$\frac{\partial}{\partial t}\theta = -\left\{(V\cdot\nabla)\theta + \dot{\sigma}\frac{\partial}{\partial\sigma}\theta\right\} + \frac{1}{c_p}\frac{\theta}{T}Q \tag{4}$$

$$\frac{1}{\pi}\frac{\partial}{\partial\sigma}\Phi = -\frac{1}{\rho} \tag{5}$$

$$\frac{\partial}{\partial t}q = -\left\{(V\cdot\nabla)q + \dot{\sigma}\frac{\partial}{\partial\sigma}q\right\} + (E-C) \tag{6}$$

Figure 1. System of Equations for GISS Model

The horizontal grid of the model has the configuration of a Mercator projection with increments in latitude of 4 degrees and in longitude of 5 degrees. With the 9 layers, this results in a total of 29,160 grid points, each having 4 points associated with it as drawn in figure 2. This way, variables of identical location in true physical space are attached to different points on the grid mesh. This placement is of great importance in maintaining the numerical stability of the integration, as well as simulating planetary flows.

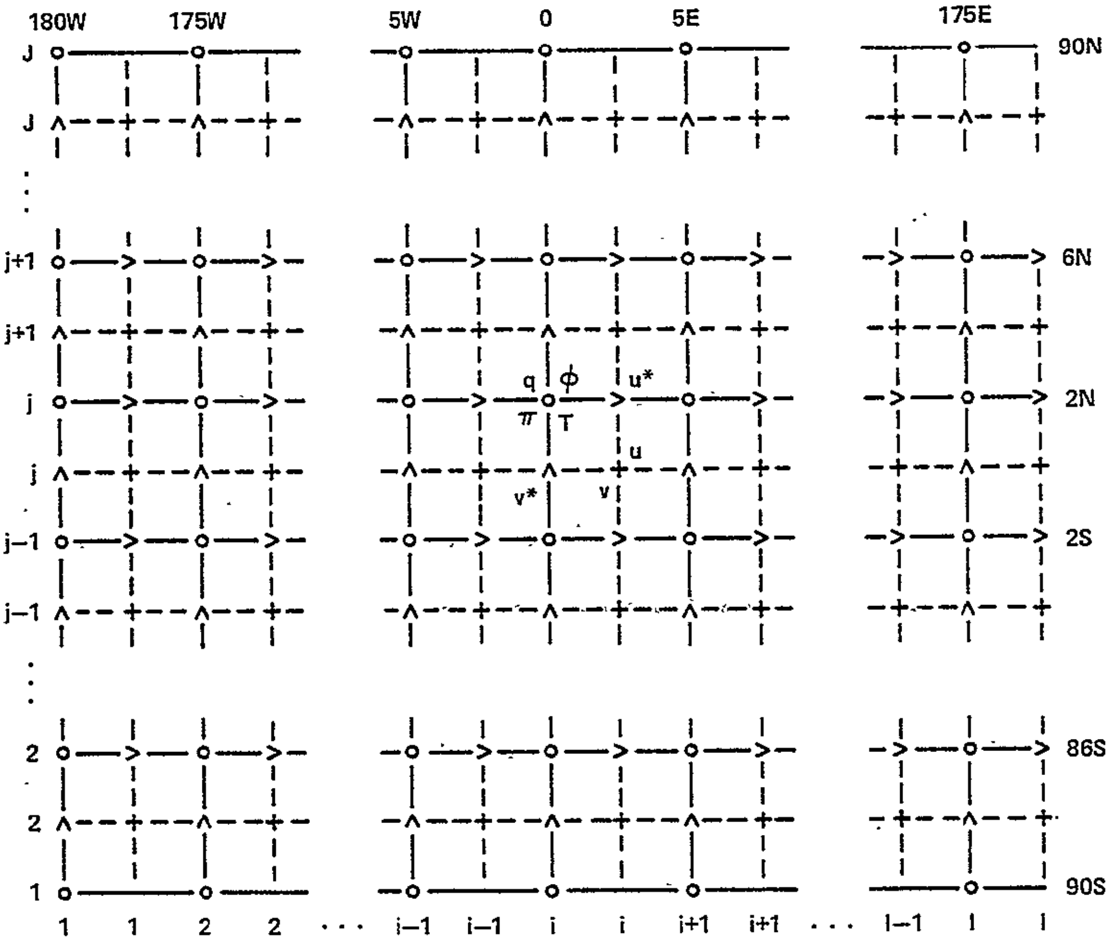


Figure 2. Finite Difference Grid for GISS Model

In older versions, the grid of the model contained an equal number of longitudinal points at all latitudes. This means that a grid square near the pole will occupy less area than a grid square near the equator. This nature of the grid was later modified to have the capability of readjusting the relative areas. The latter is known as a split grid in which the number of longitudinal points can be properly selected at each latitude.

In the prediction equations (1), (2), (4), and (6), terms other than the time derivatives on the right-hand side fall into two categories, namely, dependent terms and source terms (refer to figure 1). Specifically, Q in (4) and C and E in (6) are source terms. They arise from physical processes and are generally small in magnitude relative to those dependent terms which arise from the conservation and continuity properties of the atmosphere. Consequently, dependent terms are evaluated at every time step but source terms are not. Some source terms that come from radiation are even smaller. To conserve computations, source terms other than radiation are evaluated once every 6 time steps and radiation once every 24 time steps. The time step is chosen to be 5 minutes.

For time integration at each time step, a first estimate B, as a predictor, is obtained with the forward differencing in time

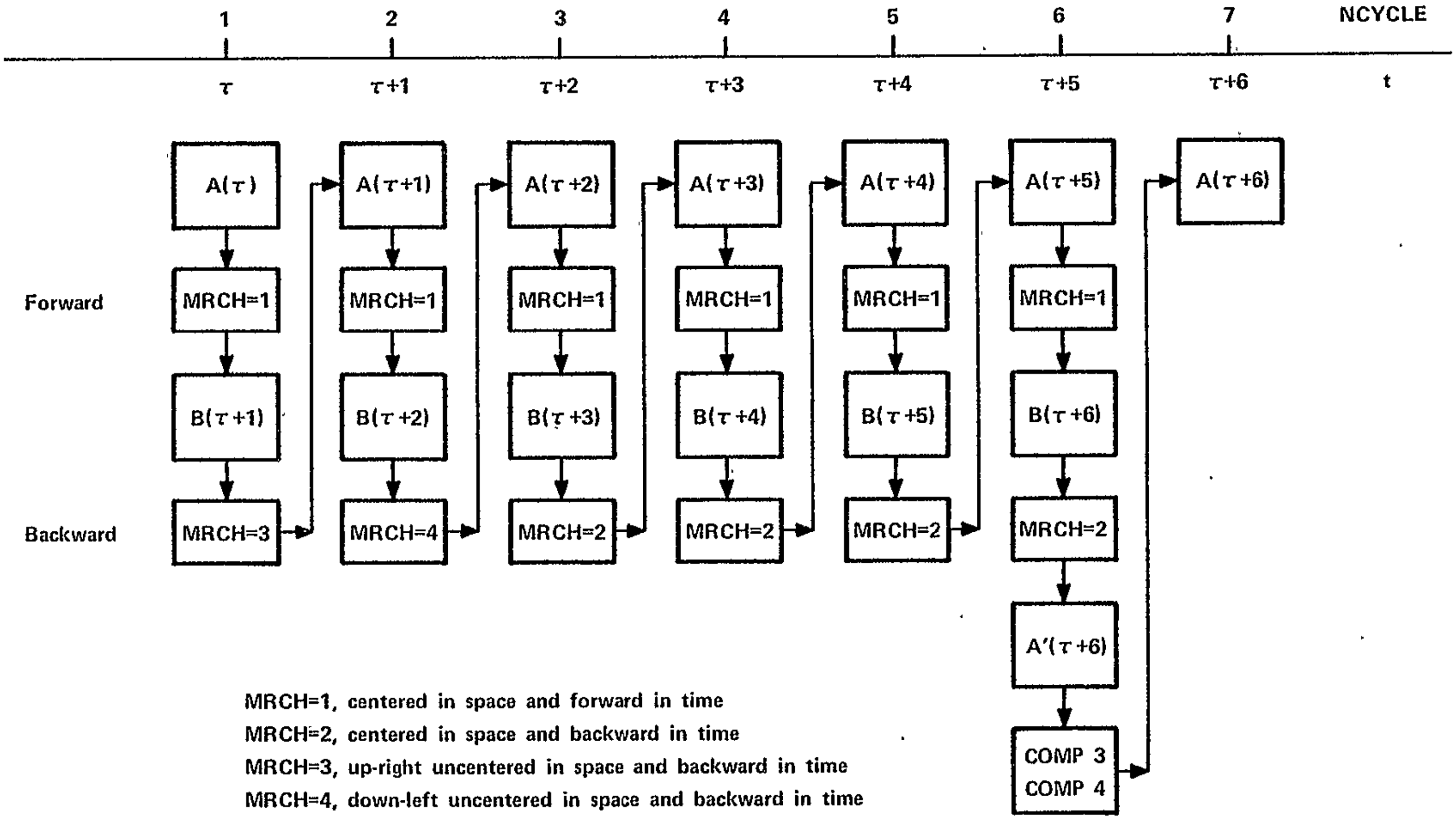
$$B(n) = A(n-1) + \Delta t * D(A(n-1))$$

where A represents the appropriate physical quantity such as one of the primary dependent variables, n is the index for a particular time step, D( ) represents the value evaluated for the derivative which is the contribution from appropriate terms on the right-hand side in the prediction equation. With such a first estimate of B(n), a second and refined estimate A(n) is then obtained with the backward differencing in time

$$A(n) = A(n-1) + \Delta t * D(B(n))$$

to serve as a corrector. This predictor-corrector cycle completes one time step of integration. The predictor and corrector are signified by the flags MRCH=1 and MRCH=2, respectively (see figure 3). In both cases, the space differencing is centered. For the first 2 time steps however, D( ) is evaluated as follows: for n=1, by up-right uncentered space differencing flagged as MRCH=3; and for n=2 by down-left uncentered space differencing flagged as MRCH=4.

Every sixth cycle of this marching process, an additional prediction cycle is imposed upon the already predicted value, with D( ) now representing only the contribution of the physical processes (but excluding that of radiation), as indicated by the box of COMP3 and COMP4 in figure 3. Every 24th cycle, the radiational contribution is included with all the physical processes mentioned above. Also, as represented by n=0, the initial quantities are given as data input to start this marching process. This marching process in time is summarized in figure 3.



MRCH=1, centered in space and forward in time
MRCH=2, centered in space and backward in time
MRCH=3, up-right uncentered in space and backward in time
MRCH=4, down-left uncentered in space and backward in time

Figure 3. Marching Sequence for GISS Model

In the actual computation, primary dependent variables do not stand by themselves. Rather, they appear in the form of pressure-area-weighted variables like πA. Accordingly, they have to be scaled by the factor $\pi(\tau)$ at the beginning of each time step and de-scaled by the factor $\pi(\tau + \Delta t)$ after the integration of the time step. Furthermore, in order to avoid reducing the time step in high latitudes, the zonal mass flux and zonal pressure force are smoothed longitudinally.

The distribution of variables over horizontal grid points adopted in the GISS model, shown in figure 2, is suitable for simulating the geostrophic adjustment. The space differencing for the nonlinear advective (dependent) terms in the equation of motion is constructed so as to maintain a constraint analogous (but not strictly equivalent) to mean square vorticity (enstrophy) conservation. In fact, the differencing for these terms reduces to the enstrophy- and kinetic-energy-conserving Jacobian scheme, when the mass flux is non-divergent. The resulting integral quasi-conservation of enstrophy as well as kinetic energy is an effective aid to preserving the shape of the energy spectrum and the area enclosed by it.

3.0 GISS MODEL VECTORIZATION

The GISS code conceptually is divided into two parts, namely, the physics portion and the dynamics portion. The physics portion of the code is found in subroutine COMP3, while the dynamics is found in subroutines COMP1 and COMP2. The division of the physics into two separate routines was done mainly as a matter of convenience, to reduce the length of the source programs and divide the workload among the team members. A byproduct of this division is, however, that COMP1 contains all of the data dependent branches (which impact the vectorization process) that appear in the dynamics calculations.

COMP1 and COMP2 are readily vectorizable routines, yielding vectors of moderate length for the model in its present state of resolution. COMP3, however, consists of a collection of physical processes which are programmed in composites of many imbedded short loops (yielding very short, inefficiently processed vectors).

The COMP3 routine in the original scalar model was called once for every grid point, as opposed to being called to process a plane or subplane of grid points. This was due to the concern for storage of temporary data on the 7600 class machines that were being used for the model. Given a massive amount of memory, COMP3 can be directly recoded to process all grid points at a single call. Thus the short DO loops, induced by the inherent physics, will be applied to groups of data points rather than a single point. This form of vectorization leaves the original computations almost exactly as it was in the original scalar model. The sheer size of the GISS model and the amount of resources needed to recode the model in the same way the 3-D implicit code was recoded makes the effort impractical under the current study contract.

Control Data was fortunate to possess a version of the GISS model which had been vectorized for the STAR-100 by David Soll and his team at the Goddard Institute for Space Studies. Since the basic programming characteristics of the STAR family and the FMP are the same, it was felt that the STAR version could be used for purposes of this report. In this model the physics processing in COMP3 is recoded to process multiple grid points together, as was discussed previously. Since the model and results of its operation have been the subject of several public disclosures (e.g. ref. 6) a decision was made to limit this investigation to two key routines which illustrate the behavior of the FMP when challenged by the GISS code.

It should be noted that although the STAR-100 version of the GISS code solves the same problem as the metric provided by Ames, and retains the same basic structure, some of the mathematics had to be revised to achieve effective vectorization of the model. This difference will become evident by examination of the AVRX and LINKHO routines, whose listings appear in appendices A and B. The other major difference between the metric provided by Ames and the STAR-100 model is the degree of resolution in each. The STAR-100 version uses a 24x16 grid while the Ames version uses a 46x72 grid. The result of this is that vector lengths in the model studied by Control Data are shorter (384 versus 2312 in most instances). The shorter vector length happens to be below the knee in the theoretical performance curve of the FMP, and thus yields considerably less than optimum results. The effect of this is that any extrapolations made for the FMP performance on the GISS model should be considered essentially 'worst case' for comparison with the model provided by Ames.

Figure 4 displays the original code for the routine AVRX, as it appeared in the Ames version of the model. Figure 5 is a listing of a version of AVRX for a climatology simulation with a course grid. This version was then adopted by David Soll for the STAR-100, the coding for which is found in figure 6, restated in figures 7 and 8 for the purposes of this discussion.

Figure 7 contains loops in J=2,5 and J=12,15 with 7(a) showing an imbedded loop on K=2,24, while 7(b) shows a sequence solely for elements at K=1 and K=25. The sequence in 7(b) can conceivably be executed in parallel with that in 7(a). If the double loop in 7(a) K=2,24 is extended to K=1,26, the whole double loop can be addressed with a single vector statement of length 26x4=104, whilst the loop in 7(b) is processed by the Scalar Unit. The cost of this technique is that, for each J, elements K=1, K=2, K=24, K=25, and K=26 must be saved beforehand, as well as K=1, K=25 updated and K=26 restored after finishing the vector operation. This updating and saving requires the use of gather and scatter operations.

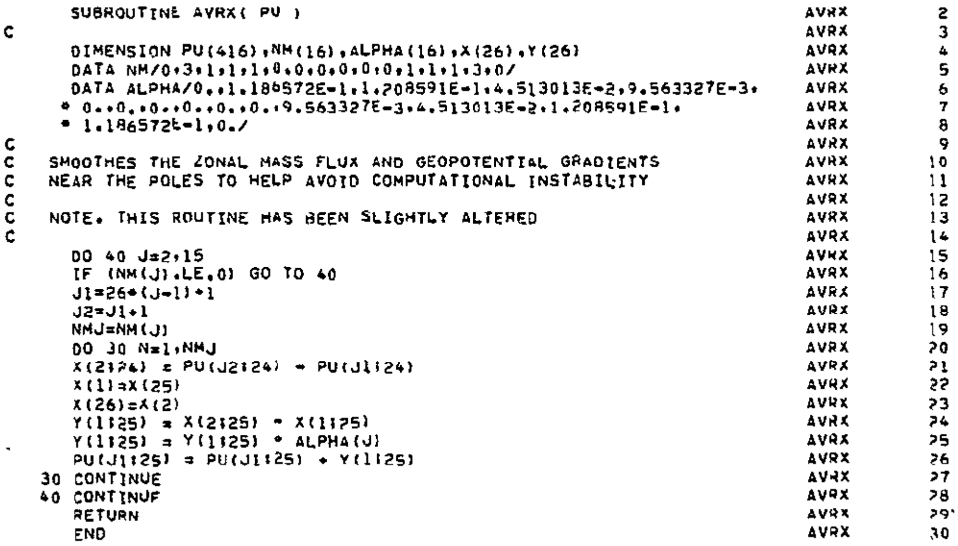
```
      SUBROUTINE AVRX
C          C D E  B L O C K  F894  A S  O F  N O V  2 9 ,  1 9 7 3
C
C     NOTE COMMON IS SPECIFIED TWICE BECAUSE THE NUMBER OF CONTINUATIONS
C     EXCEEDED THE COMPILER LIMIT
C
      REAL*4 KAPA,LAT
      C O M M O N
     *  JSP,JNP,IM,NLAY,PTROP,ISTART,JSPP1,JNPM1,FIM,NLAYM1,NLAYP1,
     *  J1,JM,KM,TAUT,IROT,MROT,TAUTHT,MROTHT,
     *  NR,JAYS(12),INCS(11),JSB,JNB,DLAT,DLON,JTEST,ITEST,
     *  DT,TAU,ITAU,XINT,IDAY,JDAY,TOFDAY,JDATE,JMONTH(2),JYEAR,NSTEP,
     *  NCYCLE,NCOMP3,NHUGAN,TAUP,TAUI,TAUE,TAUO,
     *  PI,GRAV,RGAS,KAPA,PSL,ED,FMU,NFLW,PSF,MRCH,RSDIST,SIND,COSD,
     * NC3T,MROC3T,LC3T,PROJAR,TOTABS,SOX,ISPACE,DUMMY1(9),TAUU,TIPE,
     *   MACHIN,
     *  IALTER,NNIMB,PTB,PTB2,NIMPTS,SETNIM,MINJ,MAXJ,IRAND,IRTYPE,ISS,
     *  SS,IDIAG,PERIOD,ALPHA(4),BETA,GAMMA(4),TIME0(4),KTPB,RINFLU,
     *  MODELT,LRAD(16),NMCT,NRAD(16),KTPA,JTAPE,CDX,FLAGS(10),
     *  MINI,MAXI,DUMMY2(15),TOTRMS,INST,TVRMS,INSV,TRMSU,INSU,TINT,
     *  XLABEL(20),SIG(20),DSIG(20),SIGE(21),DSIG0(19),
     *  J1PS(11),JMPS(11),J1US(11),JMUS(11),J1P,JMP,J1U,JMU,J1PV,JMPV,
     *  INC,J1PY,J1PM1,JMPY,JMPP1,JMPP1X,J1UX,J1UM1,J1UM1X,JMUX,FINC,
     *  INC2,INCH
      C O M M O N
     *      U,V,T,SH,P,TS,SHS,GT,GW,TOPOG,UT,VT,TT,SHT,PT,SPA,PV,PU,
     *  CONV,PHIS,
     *  LAT,DXU,DXP,DYU,DYP,SINL,COSL,DXYP,F,C1,DUMMY,NTAB,RHXX(46,72,9)
     *  ,RHMAX,INWRFG,TAUINW
      DIMENSION P(46,72),U(46,72,9),V(46,72,9),T(46,72,9),SH(46,72,9),
     *  TS(46,72),SHS(46,72),TOPOG(46,72),PT(46,72),UT(46,72,9),
     *  VT(46,72,9),TT(46,72,9),SHT(46,72,9),SPA(46,72),PV(46,72),
     *  PU(46,72),CONV(46,72,20),GT(46,72),GW(46,72),PHIS(46,72),
     *  Q(46,72,9,4),QT(46,72,9,4),
     *  SD(46,72,8),QS(46,72,2),US(46,72),VS(46,72),PHI(46,72),
     *  PIT(46,72),FD(46,72),
     *  LAT(46),DXU(46),DXP(46),DYU(46),DYP(46),SINL(46),COSL(46),
     *  DXYP(46),F(46),C(300),C1(300),DUMMY(72)
      EQUIVALENCE (Q(1,1,1,1),U(1,1,1)),(QT(1,1,1,1),UT(1,1,1)),
     *  (SD(1,1,1),CONV(1,1,1)),(QS(1,1,1),US(1,1),SPA(1,1)),(VS(1,1),
     *  PHI(1,1),PIT(1,1),PV(1,1)),(FD(1,1),PU(1,1)),(C(1),JSP)
      COMMON/ALBCOM/RSURF,SLBEDO(46,72),VALBFG,ITC,GSW,JALB,IALB,SMOOTH
      LOGICAL SMOOTH
      INTEGER*4 SLBEDO
      LOGICAL FLAGS,INWRFG,VALBFG
C
C END     C D E  B L O C K  F894  A S  O F  N O V  2 9 ,  1 9 7 3
C
CINCL 009
      DEFF=DYP(J1P)
      DO 40 J=J1P,JMP
      DRAT=DYP(J)/DXP(J)
      IF(DRAT.LT.1.) GO TO 40
      ALP=.125*(DRAT-1.)
      NM=DRAT
      FNM=NM
      ALPH=ALP/FNM
      DO 30 N=1,NM
      IMINC=IM-INC
      I=IM
      DO 10 IPINC=INC,IM,INC
      DUMMY(I)=PU(J,I)+ALPH*(PU(J,IPINC)+PU(J,IMINC)-PU(J,I)-PU(J,I))
      IMINC=I
   10 I=IPINC
      DO 20 I=INC,IM,INC
   20 PU(J,I)=DUMMY(I)
   30 CONTINUE
   40 CONTINUE
      RETURN
      END
```

Figure 4. Original Code for AVRX Routine

```
      SUBROUTINE AVRX                                                   0AVRX
C****                                                                   0AVRX
C**** THIS SUBROUTINE SMOOTHES THE ZONAL MASS FLUX AND GEOPOTENTIAL     0AVRX
C**** GRADIENTS NEAR THE POLES TO HELP AVOID COMPUTATIONAL INSTABILITY. 0AVRX
C****                                                                   0AVRX
C**** COARSE NINE LAYER CDE BLOCK   FEBRUARY 1976                       0AVRX
      REAL KAPA,LAT                                                     0AVRX
      COMMON JM,IM,NLAY,PTROP,JMM1,FIM,NLAYM1,NLAYP1,DLAT,DLON,         0AVRX
     *  ISTART,KM,TAUT,IROT,MROT,                                       0AVRX
     *  DT,TAU,ITAU,XINT,IDAY,JDAY,TOFDAY,JDATE,JMONTH(2),JYEAR,NSTEP,  0AVRX
     *  NCYCLE,NCOMP3,NHOGAN,TAUP,TAUI,TAUE,TAUO,MRCH,                  0AVRX
     *  PI,GRAV,RGAS,KAPA,PSL,ED,FMU,NFLW,PSF,RSDIST,SIND,COSD,RHMAX,   0AVRX
     *  COX,DUMMYC(151),                                                0AVRX
     *  XLABEL 20),SIG(20),DSIG(20),SIGE(21),DSIGO(19),                 0AVRX
     *  LAT(16),SINL(16),COSL(16),DXU(16),DXP(16),DYU(16),DYP(16),      0AVRX
     *  DXYP(16),F(16),DUMMY(24),                                       0AVRX
     *  TS(16,24),SHS(16,24),GT(16,24),GW(16,24),PHIS(16,24),           0AVRX
     *  TOPOG(16,24)                                                    0AVRX
      DIMENSION U(16,24,9),V(16,24,9),T(16,24,9),SH(16,24,9),P(16,24),  0AVRX
     *  C(300)                                                          0AVRX
      EQUIVALENCE (C(1),JM)                                             0AVRX
      COMMON U,V,T,SH,P                                                 0AVRX
      COMMON/WORK/PU(16,24)                                             0AVRX
      DO 40 J=2,JMM1                                                    0AVRX
      DRAT=DYP(2)/DXP(J)                                                0AVRX
      IF(DRAT.LT.1.) GO TO 40                                           0AVRX
      ALP=.125*(DRAT-1.)                                                0AVRX
      NM=DRAT                                                           0AVRX
      FNM=NM                                                            0AVRX
      ALPH=ALP/FNM                                                      0AVRX
      DO 30 N=1,NM                                                      0AVRX
      IM1=IM-1                                                          0AVRX
      I=IM                                                              0AVRX
      DO 10 IP1=1,IM                                                    0AVRX
      DUMMY(I)=PU(J,I)+ALPH*(PU(J,IP1)+PU(J,IM1)-PU(J,I)-PU(J,I))       0AVRX
      IM1=I                                                             0AVRX
   10 I=IP1                                                             0AVRX
      DO 20 I=1,IM                                                      0AVRX
   20 PU(J,I)=DUMMY(I)                                                  0AVRX
   30 CONTINUE                                                          0AVRX
   40 CONTINUE                                                          0AVRX
      RETURN                                                            0AVRX
      END                                                               0AVRX
```

Figure 5. Version of AVRX for Climatology

Simulation with a Course Grid



```
      SUBROUTINE AVRX( PU )
C
      DIMENSION PU(416),NM(16),ALPHA(16),X(26),Y(26)
      DATA NM/0,3,1,1,1,0,0,0,0,0,0,1,1,1,3,0/
      DATA ALPHA/0.,1.186572E-1,1.208591E-1,4.513013E-2,9.563327E-3,
     * 0.,0.,0.,0.,0.,0.,9.563327E-3,4.513013E-2,1.208591E-1,
     * 1.186572E-1,0./
C
C    SMOOTHES THE ZONAL MASS FLUX AND GEOPOTENTIAL GRADIENTS
C    NEAR THE POLES TO HELP AVOID COMPUTATIONAL INSTABILITY
C
C    NOTE. THIS ROUTINE HAS BEEN SLIGHTLY ALTERED
C
      DO 40 J=2,15
      IF (NM(J).LE.0) GO TO 40
      J1=26*(J-1)+1
      J2=J1+1
      NMJ=NM(J)
      DO 30 N=1,NMJ
      X(2;24) = PU(J2;24) - PU(J1;24)
      X(1)=X(25)
      X(26)=X(2)
      Y(1;25) = X(2;25) - X(1;25)
      Y(1;25) = Y(1;25) * ALPHA(J)
      PU(J1;25) = PU(J1;25) + Y(1;25)
   30 CONTINUE
   40 CONTINUF
      RETURN
      END
```

Figure 6. Code of Figure 5 Vectorized for the STAR-100

```
  J = 2, 5 (or J = 12,15)

  DO 20      K = 2, 24

  P(K, J) = [{P(K+1, J) – P(K, J)}

           –{P(K, J) – P(K-1, J)}]

           * A + P (K, J)

20 CONTINUE
```

(a)

```
Z = [{P(2, J) – P(1, J)}

     – {P(25, J) – P(24, J)} ]

     * A

P(1, J) = Z + P(1, J)

P(25, J) = Z + P(25, J)

P(26, J)
```

(b)

Figure 7. Restated AVRX Loops in J=2,5 and J=12,15

The code in figure 8 is executed twice. The J index takes the values 2 and 15 during these executions. This fact is highlighted in figure 8 by underlining the indices J=2, the first of which can be found to have a 15 underneath it. As in the previous example, figure 8 consists of two parts, (a) and (b). The loop shown in 8(a) can be put in a vector form with length of only 23 elements. There is no need in this case, however, to gather and scatter data in order to save and restore elements, since the large register file of the FMP can be scheduled to handle the necessary sequence of scalar load and store operations.

In addition to sequences like AVRX which cause concern for analysts wishing to optimize code for the STAR-100/FMP, another subroutine loomed as a mighty challenge for FMP vectorization. The subroutine LINKHO in the original GISS model was recoded for the Control Data FMP and yielded a sparse .020 gigaflop performance. No amount of 'local' vectorization, as was applied to the other metrics, could seem to yield any better performance. The LINKHO routine developed by David Soll was then acquired and its STAR-100 listing is included in appendix B. The methodology employed in this vectorization is amply discussed in reference 6.

$$K = 2, 24$$

$$
\begin{aligned}
P(K, \tfrac{2}{15} = \quad & [\{P(K{+}1, \underline{2}) - P(K, \underline{2})\} \\
& -\{P(K, \underline{2}) - P(K{-}1, \underline{2})\}] \\
& * A + P(K, \underline{2})
\end{aligned}
$$

(a)

$$
\begin{aligned}
Z = {} & [\{P(2, \underline{2}) - P(1, \underline{2})\} \\
& -\{P(25, \underline{2}) - P(24, \underline{2})\}] \\
& * A
\end{aligned}
$$

$$P(1, \underline{2}) = z + P(1, \underline{2})$$

$$P(25, \underline{2}) = Z + P\,(25, \underline{2})$$

(b)

Figure 8. Restated AVRX Executed for J=2 and J=15

4.0 THE MIT MODEL

The MIT model is authored principally by Cunnold, Alyea, Phillips, and Prinn and supported as part of the Climatic Impact Assessment Program by the U.S. Department of Transportation. Its objective is to simulate the distribution of ozone as effected by a simple but reasonably realistic dynamical model with some more-up-to-date photochemistry. The dynamics is simplified by the quasi-geostrophic approximation so that a time step of one hour can be used because the pronounced seasonal variations in ozone require several years of integration before a statistically steady state can be expected.

Although the heating due to the absorption of solar radiation by ozone is computed explicitly and precisely for the stratosphere, empirical representation is employed for the lower atmosphere. The latter is not appropriate for daily prediction of tropospheric weather, nor is it satisfactory as a means of predicting in a fundamental sense such properties as the typical pole-to-pole temperature difference in the troposphere. In spite of its simplicity, it fulfills its purpose to create zonal flows, large-scale eddies, and meridional circulations in the troposphere which are realistic in a statistical sense, and which can be affected by stratospheric ozone at higher elevations and can redistribute that ozone in a natural manner.

The model was used to simulate stratospheric motion patterns, meridional circulations, and ozone density over a 3-year period as a function of height and latitude, eddy transports of ozone, surface destruction of ozone, correlations of ozone with other variables, and annual cycle of columnar ozone in high latitude. It has a mission to predict ozone distribution under some perturbed conditions such as those which could conceivably be caused by NO_X from aircraft engines.

For the MIT model, the system of governing equations, shown in figure 9, is basically similar to that of the GISS model shown in figure 1. Comparing the two figures, a few differences can be noted. First, because the emphasis of the MIT model is more on the development in the upper atmosphere where the water vapor is virtually non-existent, equation (6) in figure 1 for the conservation of the specific humidity is replaced by a corresponding one for the ozone mixing ratio denoted as (6') in figure 9. The heating term Q in equation (4) will consequently no longer have any contribution from the precipitation. Rather, the radiation is the only process that plays here.

Secondly, the quasi-geostrophic approximation necessitates some reformulation of the equation of motion for prediction. A vorticity equation is first derived from the equation of motion by cross-differentiation to cancel the pressure term. In the meantime, the vorticity can be represented in terms of the geostrophic wind and, in turn, the balanced pressure which can be taken as the stream function of the horizontal velocity. Thus, equation (1) in figure 1 as the prediction equation of motion is replaced by the vorticity equation in terms of the stream function as in equation (1'), together with the balanced condition (geostrophic), equation (1'a), both in figure 9.

The vertical coordinate is $Z=-\ln P$ where P is pressure divided by 1000 mb which represents the surface pressure. In this vertical coordinate, the continuity equation takes the form shown as equation (2'), and the hydrostatic equation is given in equation (5').

Equation (3), the equation of state, remains the same and is duplicated as (3'). The thermodynamic equation (4) becomes (4') in figure 9, where Q contains the absorption of solar radiation by ozone as well as infrared cooling.

$$\nabla^2 \frac{\partial \psi}{\partial t} = -\mathrm{k} \times \nabla \psi \cdot \nabla (f + \nabla^2 \psi) - \nabla \cdot f \nabla \frac{\partial X}{\partial P} - F \tag{1'}$$

$$\nabla^2 \phi' = 2\omega \nabla \cdot \mu \nabla \psi \tag{1'a}$$

$$W = \nabla^2 \frac{\partial X}{\partial P} \tag{2'}$$

$$\frac{p}{\rho} = RT \tag{3'}$$

$$\frac{\partial T'}{\partial t} = -\mathrm{k} \times \nabla \psi \cdot \nabla T' - WS + \frac{Q}{c_p} \tag{4'}$$

$$RT' = \partial \phi' / \partial Z \tag{5'}$$

$$\frac{\partial \chi_{O_3}}{\partial t} = -\mathrm{k} \times \nabla \psi \cdot \nabla \chi'_{O_3} - W \frac{\partial}{\partial Z} \overline{\chi}_{O_3} + \frac{1}{H_o^2 P} \frac{\partial}{\partial Z} \left(K_d P \frac{\partial \chi'_{O_3}}{\partial Z} \right) + G' \tag{6'}$$

Figure 9. System of Equations for MIT Model.

The derivation of this set of equations went back to Lorenz' work in 1960 for an energetically consistent formulation of the quasi-geostrophic system (ref. 7). In order to have suitable energy invariants, all terms in the vorticity equation which involve both the rotational and the divergent part of the wind field should be omitted.

Peng in 1965 (ref. 8) was the first one to successfully perform a simple numerical experiment concerning the general circulation in the lower stratosphere with Lorenz' dynamical system. In addition, he chose the spectral method on the grounds that

- i) the variation of the Coriolis parameter can be handled easily;
- ii) there is no distortion of mass distribution;
- iii) no artificial lateral boundary conditions are necessary.

For simplicity, only wave numbers 2 and 6, in addition to wave number 0, are included to represent the extra-long wave in the stratosphere and the dominant unstable wave in the troposphere, respectively.

Following in Peng's footsteps, Clark used the same dynamical framework in 1970 (ref. 9) to model the radiative and photochemical process as suggested by Lindzen and Goody (ref. 10) for the winter stratospheric circulation with waves 0,1,2,3, and 6. Clark did not include odd nitrogen but increased his ozone absorption coefficients artifically by 40%. The integration was carried out to only 230 simulated days which is certainly not long enough to show any complete annual cycle.

The MIT model continues the work of Clark to predict ozone by simulation of a three-year period with a 25-layer dynamic model. The distributions of NO_2 and odd hydrogen deduced by McConell and McElroy (ref. 11) are used to incorporate in a simple way the chemical effect of these species. The dynamics is essentially the same as that of Clark and Peng, except that all waves up to wave number 6 were included.

Because the number of waves resolved is as low as 6, taking advantage of the tendency for larger-scale motion in the stratosphere, use of the classical interaction coefficient method is justified to handle the nonlinear terms. An exception to this is the ozone heating term Q of (4') and photochemical term G' in the rate of change of ozone mixing ratio (6').

For convenience, the two hemispheres are assumed to be geographically similar. This has the advantage that the approach to statistically stationary behavior of ozone can be examined at intervals of 6 months rather than 12 months in the presence of a pronounced annual cycle at a fixed latitude. The difference between hemispheres is a matter which the authors of the code chose to postpone. Accordingly, the orographic height used in the model is defined such that the southern hemisphere is a mirror image of the north.

The vertical domain of the integration extends from Z=0 at the surface to Z=10.13675 at the model top which corresponds to an isobaric lid on the model at a pressure of about 0.04 mb, a standard atmospheric height of 71.6 km. This height was chosen to be suitably far above the main ozone layer, high enough to include the photochemical equilibrium region and to minimize the mechanical effects on the motions below. The range in Z is divided evenly into 25 layers, each of thickness $\Delta Z=0.40574$. The values of stream function are defined at the midpoints of these layers, while vertical motion and temperature are defined at the interfaces. The coordinate Z varies almost linearly with height according to the hydrostatic relation so that ΔZ corresponds almost uniformly to a height increment of 2.89 km. This choice gives good resolution in the stratosphere.

The spectral method calls for expansion of variables in terms of spherical harmonics. Thus the representation of the stream function, for example, takes the form of equation (7) in figure 10. Equation (8) gives the structure of the spherical harmonics and (9) gives the associated Legendre Polynomial. Note that (8) is in complex form and to insure that the fields are real, it is necessary to have equations (10) and (10a) in which the asterisk denotes the complex conjugate. Equation (11) gives the vorticity.

The methodology is illustrated with a shorter version of the vorticity equation (1') having only 2 terms, namely the Coriolis term and the advection term as in equation (12). Inserting the appropriate expansions into the vorticity equation (12) and integrating the whole equation with respect to μ and λ obtains the transformed form as equation (13). The truncation used to date has M=J=6. This provides 79 degrees of freedom in each variable at each vertical level.

$$\psi(\mu,\lambda) = \sum_{m=-M}^{M} \psi_m(\mu)\, e^{im\lambda} = \sum_{m=-M}^{M} \sum_{n=|m|}^{|m|+J} \psi_{m,n} Y_{m,n} \tag{7}$$

$$Y_{m,n}(\mu,\lambda) = P_{m,n}(\mu)\, e^{im\lambda} \tag{8}$$

$$P_{m,n}(\mu) = \left[(2n+1) \frac{(n-m)!}{(n+m)!} \right]^{1/2} \frac{(1-\mu^2)^{m/2}}{2^n n!} \frac{d^{n+m}}{d\mu^{n+m}} (\mu^2-1)^n \tag{9}$$

$$\psi_{-m}(\mu) = \psi_m^*(\mu) \tag{10}$$

$$\psi_{-m,n} = (-1)^m \psi_{m,n}^* \tag{10a}$$

$$\xi = \nabla^2 \psi = - \sum_{m=-M}^{M} \sum_{n=|m|}^{|m|+J} \frac{n(n+1)}{a^2} \psi_{m,n} Y_{m,n} \tag{11}$$

$$\frac{\partial \xi}{\partial t} = - \frac{2\omega}{a^2} \frac{\partial \psi}{\partial \lambda} + \frac{1}{a^2} \left[\frac{\partial \psi}{\partial \mu} \frac{\partial \xi}{\partial \lambda} - \frac{\partial \psi}{\partial \lambda} \frac{\partial \xi}{\partial \mu} \right] + \cdots \tag{12}$$

$$\frac{d\psi_{m,n}}{dt} = \frac{2\omega m}{n(n+1)} i\psi_{m,n} - \frac{a^2}{n(n+1)} F_{m,n} + \cdots \tag{13}$$

where
$$F_{m,n} = -\frac{1}{a^4} \sum_{m_1=-M}^{M} \sum_{n_1=|m_1|}^{|m_1|+J} \sum_{m_2=-M}^{M} \sum_{n_2=|m_2|}^{|m_2|+J} i\psi_{m_1,n_1} \psi_{m_2,n_2} L_{n\,n_1 n_2}^{m\,m_1 m_2} \tag{14}$$

and the interaction coefficients $L_{n\,n_1 n_2}^{m\,m_1 m_2}$ are given by

$$L_{n\,n_1 n_2}^{m\,m_1 m_2} = \tfrac{1}{2}[n_1(n_1+1) - n_2(n_2+1)] \int_{-1}^{1} P_{m,n} \left[m_1 P_{m_1,n_1} \frac{dP_{m_2,n_2}}{d\mu} - m_2 P_{m_2,n_2} \frac{dP_{m_1,n_1}}{d\mu} \right] d\mu \tag{15}$$

for $m = m_1 + m_2$

$$L_{n\,n_1 n_2}^{m\,m_1 m_2} = 0$$

for $m \neq m_1 + m_2$

Figure 10. Some Algebra of Spectral Model

The interaction coefficients are pre-calculated and stored. To extropolate in time (prediction), the "4-cycle" version of the time-differencing scheme formulated by Lorenz (ref. 12) is used for step-wise numerical integration. Taking advantage of the quasi-geostrophic approximation, a time step of one hour is used in each cycle. Using this model, four hours of computer time was required on the IBM 360/95 to integrate one simulated year.

The ozone heating term and the photochemical term are transcendentally nonlinear, and quasi-linearization to put them in a quadratic form did not seem to be accurate enough in some cases because of the rapid dependence on temperature of some reaction rates. These were therefore evaluated at each time step by the transform method as follows:

i) transform the wave number representation of variables to physical space at points of a fixed longitude-latitude grid;

ii) compute the above terms at these physical points;

iii) transform these computed physical point values back into the spectral form;

iv) add these contributions (now in spectral form) to the remaining terms (that were calculated by the interaction coefficient method).

The longitude-latitude grid of the physical space employed in this transform method had 16 points in longitude and 15 in latitude. The latter provides an exact quadrature for the quadratic product.

The heating scheme consists of a relatively precise evaluation of heating in the stratosphere combined with the more empirical Newtonian cooling in the lower atmosphere (see figure 11). The contribution due to the absorption of solar radiation by ozone is computed explicitly by evaluating the integral of equation (17) in figure 11.

$$Q1 = (\chi_{O_3}/\beta)Q(N\sec\zeta) - c_p h(Z)T' \tag{16}$$

where $$Q(N\sec\zeta) = \int \alpha I \epsilon \exp(-\alpha N \sec\zeta)\, d\Lambda \tag{17}$$

$$Q2 = c_p h(Z)\,(T^* - T') \tag{18}$$

$$Q = Q1 + Q2 \tag{19}$$

Figure 11. Heating/Physics for MIT Model

The same kind of integral is also employed in the computation of photochemistry as in equation (21) of figure 12. For speed of computation, a table was initially evaluated by numerical procedure in integration with respect to Λ for a large range of $N(\sec\zeta)$ in equations (17) and (21). In the course of the model simulation, a table look-up is then performed in place of actually carrying out the integral when encountered.

The model photochemistry includes the Chapman reactions, the NO and NO_2 catalytic cycle, and several reactions between hydrogen and atomic oxygen. Figure 12 includes a list of the reactions used in the model.

$$G = 0.42 J_{O_2} \frac{(\chi_{O_3} J_{O_3} + \chi_{NO_2} J_{NO_2})}{0.21\, \eta\, k_0} [2(k_1 \chi_{O_3} + k_3 \chi_{NO_2}) + (k_4 \chi_{OH} + k_5 \chi_{HO_2})] \qquad (20)$$

$$\text{where } J_i = \int \alpha_i(\Lambda)\, I(\Lambda) \exp(-\alpha_i N_i \sec\zeta)\, d\Lambda \qquad (21)$$

$$i = O_3, NO_2, O_2$$

Reaction
1. $O_2 + h\nu \to 2O$
2. $O + O_2 + \gamma \to O_3 + \gamma$
3. $O_3 + h\nu \to O_2 + O$
4. $O + O_3 \to 2O_2$
5. $NO + O_3 \to NO_2 + O_2$
6. $NO_2 + O \to NO + O_2$
7. $NO_2 + h\nu \to NO + O$
8. $OH + O \to O_2 + H$
9. $HO_2 + O \to O_2 + OH$

Figure 12. Ozone Generation/Chemistry for MIT Model

5.0 SPECTRAL CODE ANALYSIS

The MIT code consists of two parts/programs, namely STRAT1 and STRAT2. The primary purpose of STRAT1 is, among other things, to establish an appropriate "initial" condition for the long-term climate simulation, with full dynamics and chemistry, which is coded in STRAT2. The main concern therefore, in this analysis, will be with the program STRAT2 only.

The code is, in general, quite well structured in that it is very modular. As can be seen in figure 8, the execution of the program STRAT2 is essentially a double loop in which one subroutine is called after another. Hence, figure 13 can also be taken as a flowchart in the sense that each subroutine represents a block for a task.

Much of the code is structured in loops. In fact, a few subroutines are composed of only one single, simple loop as displayed in appendix C. To vectorize these is a rather trivial matter.

The subroutines in appendix D have been recoded, from the standpoint of syntax only, so that they can be easily followed as well as verified. This could be considered as a pre-vectorization pass to show an intermediate step on the way to complete recoding and restructuring of data. This is discussed at greater length later in this report. In the original code the indices of arrays are managed/manipulated explicitly by the programmer. In the recoded version, use has been made of the array structure implied by the FORTRAN DIMENSION declarations in terms of the ordering of elements. This brings the logic of most routines into much better perspective, i.e., in simpler

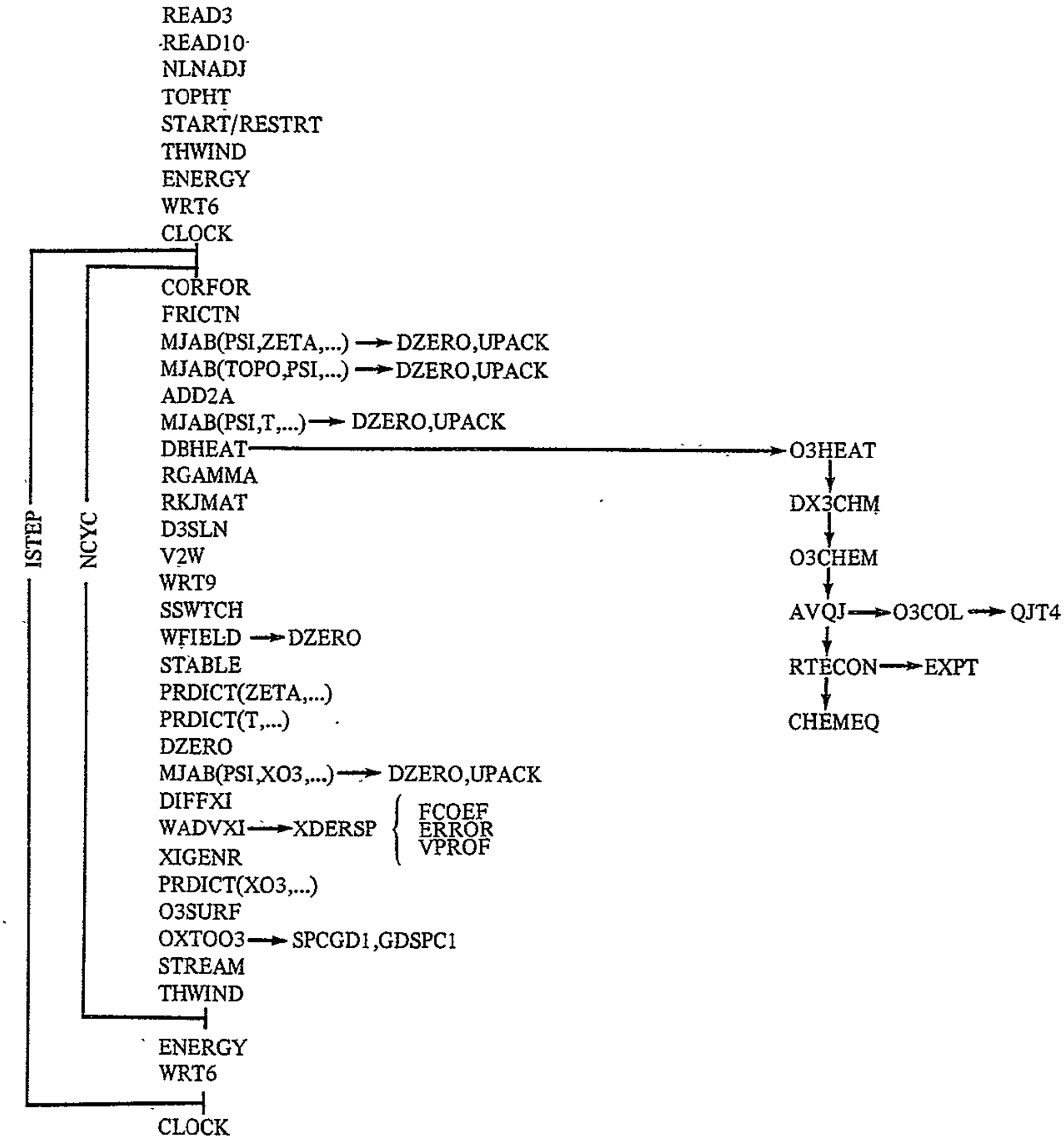


Figure 13. Program Flow of MIT Model

structures of loops whose vectorization can be effected readily. As a matter of fact, it exposed not only the possibility/feasibility, but also the advantage to restructure the data block in the entire program.

In the process of recoding, an effort was made to pull out portions of code that have to be executed/initialized only once. These are grouped together under an entry point with the same name as that of the routine except appended with a digit 0. In addition, a letter V was added as a prefix to the original name of each routine that was recoded, as a name for the recoded routine. However, the recoded version is not necessarily in vectorized form.

The discussion of the spectral code will start with the simplest routine, DZERO.

```
      SUBROUTINE DZERO(A)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,NRTP,
     1    LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      DIMENSION A(1)
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 100 I=IL,IH
 100  A(I)=0.D0
      RETURN
      END
```

This routine is used to zero out a block in the memory for initialization. The construct is straightforward; it consists of a single, simple loop 100 which can be put easily in one vector instruction as

AD=0

where AD is the descriptor representing the vector A(I),I=IL,IH. However, since each of the FMP vector pipes has two paths, it would be wise to break the vector AD into two halves, recoded as follows:

```
      SUBROUTINE VDZERO(A)
      COMMON /CONSTS/ INDEX, . . .
      DIMENSION A(1)

      DYNAMIC AD1,AD2

 9001 AD1=0.
      AD2=0.
      RETURN

      ENTRY DZEROO
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      L=IH-IL+1
      L2=L/2
      DEFINE (AD1, A(IL:IL+L2-1))
      IL2=IL+L2
      LL=LL-L2
      DEFINE (AD2, A(IL2;IL2+LL-1))
      RETURN
      END
```

In the program, ILEV1=2, ILEV2=25, and NVREAL=79. Consequently, here L=1896, L2=948 and LL=948. With the rate of 8 results (in 64-bit mode) per cycle, it takes 119=(948+4)/8 cycles for both vectors AD1 and AD2 to pass through the pipes. With the start-up of 6 cycles, 15.17 results per cycle are obtained, or 0.95 gigaflop. This takes advantage of the multiple paths of the FMP vector pipes, but not the capability of performing multiple operations in each vector pipe. This is exploited in the following.

```
      SUBROUTINE ADD2A
      COMMON P(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,NRTP,
     1    LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      K=ILEV2
      DO 200 ITMS=ILEV1,ILEV2
      IH=K*NVREAL
      IL=(K-1)*NVREAL+1
      DO 100 I=IL,IH
      J=I-NVREAL
      P(I)=.5*(P(J)+P(I))
 100  CONTINUE
      K=K-1
 200  CONTINUE
      RETURN
      END
```

This is another example of straightforward vectorization. The vector length is much shorter. The vector code is as follows:

```
      SUBROUTINE VADD2A
      COMMON P(2366)
      COMMON /CONSTS/ INDEX, . . .

      DYNAMIC QD,PD(26),PD1,PD2

      DO 150 K=1,KK,2
      K1=K+1
      K2=K+2
 9002 QD=PD(K1)+PD(K)
      PD(K1)=PD(K2)+PD(K1)
      PD(K)=QD
 150  CONTINUE
 9003 PD1=.5*PD1
      PD2=.5*PD2
      RETURN

      ENTRY ADD2A0
      IL=(ILEV2-1)*NVREAL+1
      KK=ILEV2-ILEV1+1
      DO 50 K=1,KK
      DEFINE (PD(K),P(IL:IL+NVREAL-1))
      IL=IL-NVREAL
 50   CONTINUE
      L=ILEV2*NVREAL
      L2=L/2
      DEFINE (PD1,P(1:L2))
      IL1=L2+1
      LL=L-L2
      DEFINE (PD2,P(IL1:IL1+LL-1))
      RETURN
      END
```

Notice that in loop 150 there seems to be a temporary QD. In fact, this temporary is not necessary during execution, as the FMP is equipped with 4 read ports and 2 write ports which can operate simultaneously. The appearance of the temporary QD is therefore merely for the sake (or convenience) of presentation to avoid any confusion (or ambiguity). Another point worth mentioning here is that in loop 150, the FMP is processing 2 vectors at a time, taking advantage of the parallel nature of the FMP vector pipes in performing multiple operations.

For NVREAL=79, it takes 16=6+(79+4)/8 cycles to complete the vector statement 9002. Also, 2 cycles are required for the Scalar Unit to update the indices K1 and K2. With KK=25, loop 150 takes 234=18*13 cycles. On the other hand, L=1975 and LL=988, statement 9003 takes 130=6+(988+4)/8 cycles. There are a total of 6050=(3*79+5)*25 flops. This therefore achieves 1.04 gigaflops.

```
      SUBROUTINE PRDICT(Y,Z,FY,N)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,NTRP,
     1    LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      DIMENSION Y(1),Z(1),FY(1)
      DATA IFLG/0/
      IF (IFLG.GT.0) GO TO 100
      DTINV=1.D+0/DT
      DDTINV=1.D+0/(NCYC*DT)
      IFLG=100
  100 CONTINUE
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      A=-(N-1)*DDTINV
      B=DTINV+A
      B=1.D+0/B
      IF (N.GT.1) GO TO 300
      DO 150 I=IL,IH
  150 Z(I)=B*FY(I)
      GO TO 400
  300 CONTINUE
      DO 350 I=IL,IH
  350 Z(I)=B*(A*Z(I)+FY(I))
  400 DO 450 I=IL,IH
  450 Y(I)=Y(I)+Z(I)
      RETURN
      END
```

The main code consists of loops 150, 350, and 450. All three are simple loops and can easily be put in vector form. For example, loop 450 will take the form

YD=YD+ZD

where YD and ZD represent the vectors Y(I) and Z(I), respectively, as initialized in the following. Since the FMP vector pipes are capable of performing multiple operations in parallel, it is not advantageous to separate loop 150 from loop 450 for the case of N=1. Therefore it should be recoded as follows:

```
      SUBROUTINE VPRDICT(Y,Z,FY,N)
      COMMON /CONSTS/ INDEX, . . .
      DIMENSION Y(1), . . .

      DIMENSION BB(4),CC(4)
      DYNAMIC ZD,YD,FYD,ZD1,ZD2,YD1,YD2

      IF (N.LE.1) GO TO 200
 9004 ZD=CC(N)*ZD+BB(N)*FYD
 9005 YD1=YD1+ZD1
      YD2=YD2+ZD2
      RETURN
  200 CONTINUE
 9006 ZD=BB(1)*FYD
      YD=YD+ZD
      RETURN

      ENTRY PRDICTO
      DTINV=1.D0/DT
      DDTINV=1.D0/(NCYC*DT)
      DO 1 I=1,NCYC
      AA=(1-I)*DDTINV
      BB(I)=1.D0/(AA+DTINV)
    1 CONTINUE
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      L=IH-IL+1
      DEFINE (ZD,Z(IL:IL+L-1))
      DEFINE (YD,Y(IL:IL+L-1))
      DEFINE (FYD,FY(IL:IL+L-1))
      L2=L/2
      DEFINE (YD1,Y(IL,IL+L2-1))
      DEFINE (ZD1,Z(IL:IL+L2-1))
      IL1=IL+L2
      LL=L-L2
      DEFINE (YD2,Y(IL1:IL1+LL-1))
      DEFINE (ZD2,Z(IL1:IL1+LL-1))
      RETURN
      END
```

For ILEV2=25 and NVREAL=79, the vector length is L=1975, and it will take 247=(1975+4)/8 cycles to pass through the pipes. With a startup of 6 cycles, it takes 383 cycles for 7910 flops, or 1.29 gigaflops, in the case of N not equal to 1. When N=1, 0.98 gigaflop is attained as there are 3960 flops for 253 cycles.

One of the important points is that in the case of N=1, the whole routine can be completed in one single pass of the pipes, thereby doubling the efficiency. Since N will be less than or equal to 4, the saving should be quite noticeable.

Statement 9004 is seen to take full advantage of the multiple-operations capability of the FMP vector pipes.

```
      SUBROUTINE RTECON(LEV)
      COMMON /CHEM/ XNEUT(26),TEMP(6240),XK1(240),XK3(240),DOHL(240),
     1    TABEXP(5000,2)
      COMMON /FTCST/ NLON,NLAT,NGRID
      COMMON /CHMCON/ XK1P(26),XK3P(26),ACTEN1,ACTEN2,XDOHP(26)
      DIMENSION T(570)
      TLOW=.25605/6.5536
      J=NGRID*(LEV-1)
      JP=J+1
      K=0
      DO 100 LAT=1,NLAT
      DO 100 LON=1,NLON
      J=J+1
      K=K+1
      T(K)=TEMP(J)
      IF (T(K).LT.TLOW) T(K)=TLOW
  100 CONTINUE
      CALL EXPT(TABEXP,T,NGRID)
      L=0
      K=0
      DO 200 LAT=1,NLAT
      DO 200 LON=1,NLON
      L=L+1
      K=K+1
      XK1(K)=XK1P(LEV)*T(L)
      L=L+NGRID
      XK3(K)=XK3P(LEV)*T(L)
      DOHL(K)=XDOHP(LEV)*T(L)
      L=L-NGRID
  200 CONTINUE
      RETURN
      END
```

In both loops 100 and 200, the formal indices are LON and LAT but the true indices are J, K; and L. This fact is not detected by the compiler and therefore these loops are considered uncollapsible for the purpose of vectorization. Incidently, this type of loop appears throughout the entire program. It has the virtue of ease in managing the loop.

```
      SUBROUTINE VRTECON(LEV)
      COMMON /CHEM/ . . .
      COMMON /FTCST/ . . .
      COMMON /CHMCON/ . . .
      DIMENSION T(570)

      BIT C(4),CD
      DYNAMIC CD,TEMPD(25),TD,XK1D,XK3D,DOHLD,TDL

 9011 TD=TEMPD(LEV)
      CD=TEMPD(LEV).LT.TLOW
 9012 IF (CD) TD=TLOW

      CALL EXPT(TABEXP,T,NGRID)

 9013 XK1D=XKIP(LEV)*TD
 9014 XK3D=XK3P(LEV)*TDL
      DOHLD=XDOHP(LEV)*TDL
      RETURN

      ENTRY RTECON0
      TLOW=.25605/6.5536
      MGRID=NLAT*NLON
      DEFINE (TD,T(1:MGRID))
      DEFINE (TDL,T(MGRID+1:MGRID))
      DEFINE (XK1D,XK1(1:MGRID))
      DEFINE (XK3D,XK3(1:MGRID))
      DEFINE (DOHLD,DOHL(1:MGRID))
      JP=1
      DO 1 I=ILEV1,LEV2
      DEFINE (TEMPD(I),TEMP(J:J+MGRID-1))
      J=J+NGRID
 1    CONTINUE
      RETURN
      END
```

There are 1680=7*16*15 flops. The statements 9011, 9012, 9013, and 9014 are vector instructions of length 240. Each of these will need 36=6+(240+4)/8 cycles. Therefore, the rate is 1.47 gigaflops.

Worth noting here is that statements 9013 and 9014 cannot be combined since the FMP vector pipes cannot perform more than 2 multiplications in parallel. Half of statement 9014 could be moved to statement 9013 but in that case, all 4 read ports would be put to use, and the total system would be busier. The way the above is coded, only 2 read ports would be used in statement 9013 and 3 read ports in statement 9014.

The data in the vector TEMP is prepared in the routine O3HEAT which calls DX3CHM, which calls O3CHEM, which calls RTECON. Therefore, statements 9011 and 9012 could have been rescheduled.

The routine RTECON is called by the routine O3CHEM in a loop. For greater efficiency, this structure should be modified.

```
      SUBROUTINE CHEMEQ(DUMMY,LEV)
      COMMON /SPECIE/ XO3(6240),CNO3(6240),XNO2(330)
      COMMON /FTCST/ NLON,NLAT,NGRID
      COMMON /WORKBK/ AVJO3(5280),AVJO2(5280),AVQO3(5280)
      COMMON /CHEM/ XNEUT(26),TEMP(6240),XK1(240),XK3(240),DOHL(240)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2
      COMMON /O3OX/ O3XFAC(2400),O3XCON(2400)
      DIMENSION DXO3DT(1),nUMMY(1)
      EQUIVALENCE (AVJO3(1),DXO3DT(1))
      J=(LEV-1)*NGRID
      INO2=(LEV-1)*NLAT
      K=0
      FAC=9.1E-12*XNEUT(LEV)*6856.8
      DO 300 LAT=1,NLAT
      INO2=INO2+1
      DO 300 LON=1,NLON
      J=J+1
      K=K+1
      IF (AVJO3(J).GT.1.E-20) GO TO 200
      O3XFAC(J)=1.
      O3XCON(J)=0.
      DXO3DT(J)=0.
      GO TO 300
  200 A=XK1(K)
      B=DOHL(K)+XNO2(INO2)*XK3(K)
      C=AVJO2(J)
      D=(B-1.)*C
      B=B*AVJO3(J)+A*C
      C=D
      A=A*AVJO3(J)
```

```
      O3XFAC(J)=1.+XK3(K)*AVJO3(J)/FAC
      O3XCON(J)=AVJO2(J)*XK3(K)/FAC
      D=B*B-4*A*C
      IF (D.LE.0.) D=0.
      X=(-B+SQRT(D))/2.*A
      XO3(J)=XO3(J)*O3XFAC(J)+O3XCON(J)
      Y=XO3(J)-2.5*DUMMY(K)
      IF (Y.GT.0.) GO TO 299
      XO3(J)=X*O3XFAC(J)+O3XCON(J)
      DXO3DT(J)=0.
      GO TO 300
  299 DXO3DT(J)=AVJO2(J)-DUMMY(K)
  300 CONTINUE
  500 CONTINUE
      RETURN
      END
```

This is another routine that has similar structure to the subroutine RTECON and is also called by O3CHEM in a loop just like RTECON. Here, the vector temporaries A, B, C, and D will be made to streamline the vectorization by using dynamic arrays.

```
      SUBROUTINE VCHEMEQ(DUMMY,LEV)
      COMMON /SPECIE/ . . .
      COMMON /FTCST/ . . .
      COMMON /WORKBY/ . . .
      COMMON /CHEM/ . . .
      COMMON /CONSTS/ . . .
      COMMON /030X/ . . .
      DIMENSION DX03DT . . .
      EQUIVALENCE (AVJ03 . . .

      DYNAMIC A,B,C,D,X,Y
      BIT E(4),ED,Z(4),ZD,Z1(4),Z1D
      DYNAMIC DOHLD,XK3D,AVJ02D,AVJ03D,03XFACD,03XCOND,DUMMYD,X03D,
     1    DX03DTD
      DYNAMIC XXNO2D,ED,ZD,Z1D
      DIMENSION XXNO2(16,15),XFAC(25),03XFACD(25),03XCOND(25),
     1    DX03DTD(25)
      DIMENSION AVJ03D(25),AVJ02D(25),X03D(25)

      INO2=(LEV-1)*NLAT
      DO 49 I=1,NLAT
      XXNO2(*,I)=XNO2(I+INO2)
49.   CONTINUE
      Z1D=AVJ03D(LEV).LE.1.E-20
      Y=2.5*DUMMYD
      B=DOHLD+XXNO2D*XK3D
      C=B*AVJ02D(LEV)
      03XFACD(LEV)=1+XK3D*AVJ03D(LEV)*XFAC(LEV)
      B=B*AVJ03D(LEV)+XK1D*AVJ02D(LEV)
      A=XK1D*AVJ03D(LEV)
      C=(C-AVJ02D(LEV))*A
      03XCOND(LEV)=AVJ02D(LEV)*XK3D*XFAC(LEV)
      D=B*B-4*C
      ED=D.LT.0
      X=X03D(LEV)*03XFACD(LEV)+03XCOND(LEV)
      ZD=X.GT.Y
      DX03DTD(LEV)=0
      IF (ED) D=0
      X=(SQRT(D)-B)*.5
      X=X/A
      IF (ZD) DX03DTD(LEV)=AVJ02D(LEV)-DUMMYD
      X=X03D(LEV)
      IF (Z1D) 03XFACD(LEV)=1
      03XCOND(LEV)=0
      IF (.NOT.Z1D) X03D(LEV)=X*03XFACD(LEV)+03XCOND(LEV)
      RETURN

      ENTRY CHEMEQ0
      MGRID=NLAT*NLON
      DO 24 I=ILEV1,ILEV2
      XFAC(I)=9.1E-12*6856.8*XNEUT(I)
      J=(LEV-1)*NGRID+1
      DEFINE (03XFACD(I),03XFAC(J:J+MGRID-1))
      DEFINE (03XCOND(I),03XCON(J:J+MGRID-1))
      DEFINE (DX03DTD(I),DX03DT(J:J+MGRID-1))
      DEFINE (AVJ03D(I),AVJ03(J:J+MGRID-1))
      DEFINE (AVJ02D(I),AVJ02(J:J+MGRID-1))
      DEFINE (X03D(I),X03(J:J+MGRID-1))
24    CONTINUE
      DEFINE (XK3D,XK3(1:MGRID))
      DEFINE (XXNO2D,XXNO2(*,1))
      DEFINE (XK1D,XK1(1:MGRID))
      DEFINE (DUMMYD,DUMMY(1:MGRID))
      DEFINE (DOHLD,DOHL(1:MGRID))
      DEFINE (ZD,Z(1:MGRID))
      DEFINE (Z1D,Z1(1:MGRID))
      DEFINE (ED,E(1:MGRID))
      RETURN
      END
```

This routine is relatively lengthy. A number of things can be found here. The dynamic arrays have illustrated themselves. In fact, the condition vectors ED, ZD, Z1D may be put in the form of dynamic arrays to save some syntactical handling.

Loop 49 is an example of broadcasting that creates a longer vector from a shorter one so that the former is compatible with other vectors.

In addition to vectorization, the order of the computations has been somewhat rearranged to avoid delay/conflict as necessitated by waiting for operands which would have been the result of the previous instruction. This is, however, not completely successful as can be seen in the computation of the quadratic root X. Since 8 results can be produced per cycle, and it takes 30 cycles for the first result to come out of the pipe, the vector length has to be longer than 240=8*30 to avoid any wait. Interestingly, in this case, NLON=16 and NLAT=15, the vector length is 240=15*16 which barely avoids a wait.

A square root function is included in this routine; it will not be discussed except to remark that it can be approximated with some rational function. For the purpose of timing, it will be treated as a known quantity.

There are 7221=6+15*(1+30*16) flops. It takes 16 passes of the FMP vector pipes. With the vector length of 240, a rate of 0.78 gigaflop is achieved in this routine.

In this routine, there are two data dependent paths. A strategy has been adopted to perform all the operations but to

C-5

store only the selected components. Another approach is to compress out appropriate components before operations and insert them back after computations. When the vector length is rather short, the latter is not used.

Worth noting however, is that things may be different if the loop in which this routine is called is expanded. This will be pursued further in the following.

```
      SUBROUTINE O3CHEM
      COMMON /SPECIE/ XO3(6240),CNO3(6240),XNO2(330),D4(5340),
     1    XNEVEN(176),XNODD(176)
      COMMON /FTCST/ NLON,NLAT,NGRID
      COMMON /QJBLK/ NZJ,L103,COL03(26),LEVPCM,LEVDYN
      COMMON /O3OX/ O3XFAC(2400),O3XCON(2400)
      COMMON /CHEM/ XNEUT(26),TEMP(6240),XK1(240),XK3(240),DOHL(240)
      COMMON /WORKBK/ AVJO3(5280),AVJO2(5280),AVQO3(5280)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,NRTP,
     1    LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /MNCON/ SD,CD,SDXN
      DIMENSION DUMMY(240),DXO3DT(1)
      EQUIVALENCE (AVJO3(1),DXO3DT(1))
      NH=(NLAT+1)/2
      J=0
      L=-NLAT
      DO 650 LEV=1,NZJ
      L=L+NLAT
      DO 640 LAT=1,NH
      I=L+LAT
      J=J+1
      IP=L+NLAT-LAT+1
      A=XNODD(J)*SDXN
      XNO2(I)=XNEVEN(J)+A
      XNO2(IP)=XNEVEN(J)-A
  640 CONTINUE
  650 CONTINUE
      CALL AVQJ
      ILEV1=LEVDYN
      ILEV2=NZJ
      J=(ILEV1-1)*NGRID
      INO2=(ILEV1-1)*NLAT
      DO 500 LEV=ILEV1,NZJ
      K=0
      CALL RTECON(LEV)
```

```
      DO 300 LAT=1,NLAT
      INO2=INO2+1
      DO 300 LON=1,NLON
      J=J+1
      K=K+1
      X=XK1(K)*XO3(J)+XK3(K)*XNO2(INO2)+DOHL(K)
      Y=AVJO3(J)*XO3(J)+AVJO2(J)
  300 DUMMY(K)=X*Y
      IF (LEV.GT.LEVPCM) GO TO 295
      CALL CHEMEQ(DUMMY,LEV)
      GO TO 500
  295 J=J-NGRID
      DO 299 K=1,NGRID
      J=J+1
  299 DXO3DT(J)=AVJO2(J)-DUMMY(K)
  500 CONTINUE
      RETURN
      END
```

A number of observations are in order. As noted before, in loop 500, routines RTECON and CHEMEQ are called. IT would be desirable to incorporate these routines into the loop to avoid any subroutine call. In addition, the loop should be broken into two parts, namely, one for LEV=ILEV1,LEVPCM and one for LEV=LEVPCM+1,NZJ. In fact, this division can be implemented only for the second half to avoid a conditional branch.

The broadcast which was done in the routine CHEMEQ is seen here too. The double loop 640 and 650 seems to be needed only once.

In fact, the mission of routine O3CHEM is to obtain the vector DXO3DT. In the process, intermediates such as X, Y, and DUMMY are computed; they actually play a role of dynamic temporaries. To conserve the storage space requirement, they were originally dimensioned with a moderate length for the available computing machinery. It has been said that one can buy computer time but not computer memory. The FMP offers memory size that is unthinkable to the "old timers", in addition to its revolutionary computational capability. Therefore, it is not necessary to continue being stingy in the use of memory space, though there is no point in wasting it either. As longer vectors would be preferred by the FMP for performance, it is wiser to increase the dimension for the intermediates X, Y, and DUMMY, etc. in order to avoid calling routines RTECON and CHEMEQ in loop 500.

```
      SUBROUTINE VO3CHEM
      COMMON /SPECIE/ . . .
      COMMON /FTCST/ . . .
      COMMON /QJBLK/ . . .
      COMMON /O3OX/ . . .
      COMMON /CHEM/ . . .
      COMMON /WORKBK/ . . .
      COMMON /CONSTS/ . . .
      COMMON /MNCON/ . . .
      DIMENSION . . .
      EQUIVALENCE . . .

      DYNAMIC ZK1,ZK3,TOH,DUM,ZNO
      DIMENSION YNO(16,15),ZK1P(240,25),ZK3P(240,25),ZTOH(240,25),
     1    ZFAC(240,25)
      DIMENSION ZNO(240,25)
      DYNAMIC A,B,C,D,X,Y

      CALL AVQJ
```

```
      T=TEMPD
      E=TEMPD.LT.TLOW
      YY=AVJO3D2*XO3D2+AVJO2D2
      IF (E) T=TLOW

      CALL EXPT(TABEXP,T, . . .

9020  ZK1=ZK1P*T
      ZK3=ZK3P*TL
      TOH=ZTOH*TL
      DUM=ZK1*XO3D2+ZK3*ZNO
      DUM=YY*(DUM+TOH)
9021  DXO3DTD1=AVJO2D1-DUMMYD1
      DXO3DTD3=AVJO2D3-DUMMYD3
      Z1=AVJO3D.LE.1.E-20
      Y=2.5*DUM
      B=TOH+ZNO*ZK3
      C=B*AVJO2D
      O3XFACD=1+ZK3*AVJO3D*ZFAC
      B=B*AVJO3D+ZK1*AVJO2D
      A=ZK1*AVJO3D
      C=(C-AVJO2D)*A
      O3XCOND=AVJO2D*ZK3*ZFAC
      D=B*B-4*C
      E=D.LT.0
      X=XO3D*O3XFACD+O3XCOND
      Z=X.GT.Y
      DXO3DTD=0
      IF (E) D=0
      X=(SQRT(D)-B)*.5
      X=X/A
      IF (Z) DXO3DTD=AVJO2D-DUM
      X=XO3D
      IF (Z1) O3XFACD=1
      O3XCOND=0
      IF (.NOT.Z) XO3D=X*O3XFACD+O3XCOND
      RETURN

      ENTRY O3CHEMO
      J=(LEVDYN-1)*NGRID+1
      L2=(NZJ-LEVDYN+1)*NGRID
      DEFINE (TEMPD,TEMP(J;J+L2-1))
      DEFINE (XO3D2,XO3(J;J+L2-1))
      DEFINE (AVJO3D2,AVJO3(J;J+L2-1))
      DEFINE (AVJO2D2,AVJO2(J;J+L2-1))
      L1=(LEVPCM-LEVDYN+1)*NGRID
      DEFINE (AVJO2D,AVJO2(J;J+L1-1))
      DEFINE (AVJO3D,AVJO3(J;J+L1-1))
      DEFINE (O3XFACD,O3XFAC(J;J+L1-1))
      DEFINE (O3XCOND,O3XCON(J;J+L1-1))
      DEFINE (DXO3DTD,DXO3DT(J;J+L1-1))
      DEFINE (XO3D,XO3(J;L1))
      LL=(NZJ-LEVPCM)*NGRID
      JJ=(LEVPCM-1)*NGRID+1
      LL2=LL/2
      DEFINE (DXO3DTD1,DXO3DT(JJ;JJ+LL2-1))
      DEFINE (AVJO2D1,AVJO2(JJ;JJ+LL2-1))
      DEFINE (DUMMYD1,DUM(JJ;JJ+LL2-1))
      LL3=LL-LL2
      JJ3=JJ+LL2
```

```
      DEFINE (DXO3DTD3,DXO3DT(JJ3:JJ3+LL3-1))
      DEFINE (AVJO2D3,AVJO2(JJ3:JJ3+LL3-1))
      DEFINE (DUMMYD3,DUMMY(JJ3:JJ3+LL3-1))
      NH=(NLAT+1)*.5
      J=0
      L=-NLAT
      DO 651 LEV=1,NZJ
      L=L+NLAT
      DO 641 LAT=1,NH
      J=J+1
      INX(J)=L+LAT
      IPX(J)=L+NLAT-LAT+1
641   CONTINUE
651   CONTINUE
      L0=NH*NZJ
      XNO2(INX)=XNEVEN(1:L0)+XNODD(1:L0)*SDXN
      XNO2(IPX)=XNEVEN(1:L0)-XNODD(1:L0)*SDXN
      INO2=(LEVDYN-1)*NLAT
      DO 41 I=1,NLAT
      YNO(*,I)=XNO2(I+INO2)
41    CONTINUE
      DO 42 I=LEVDYN,NZJ
      ZK1P(*,I)=XK1P(I)
      ZK3P(*,I)=XK3P(I)
      ZTOH(*,I)=XDOHP(I)
      ZFAC(*,I)=
      ZNO(*,I)=YNO(*)
42    CONTINUE
      RETURN
      END
```

The statement 9021 replaces the loop 299. Since it consists of one single operation, it is broken into 2 halves for parallel processing. The vector length in this case is 1920=16*15*16/2 and it takes 246=6+(1920+4)/8 cycles for statement 9021 to complete.

Statement 9020 can be considered similarly. The vector length here is 3000=16*15*25/2; it takes 381=6+(3000+4)/8 cycles to complete.

After statement 9021 are codes from the routine CHEMEQ except that the vectors are 9 times longer. Each therefore needs 276=6+(16*15*9+4)/8 cycles. Except for statement 9020, the vector lengths for lines before statement 9021 are 6000=16*15*25.

The code after statement 9021 needs 4140=276*15 cycles, while the code before needs 4161=756*5+381 cycles. A grand total of 138,042 flops is required. With 8547 cycles, it yields 1.01 gigaflops.

Loops 41 and 42 are for broadcasting. Loops 641 and 651 prepare the index vectors INX and IPX which are used to scatter XNO2 using indirect addressing.

```
      SUBROUTINE FFTFOR(DATARL,DATAIM)
      COMMON /FFT/ WP(7,7,15),W(2,7),NTRANS(16),NNN,NN,LR1,NLATHF,
     1     NCPAR(7),LOGN
      COMMON /FTCST/ N
      DIMENSION DATARL(1),DATAIM(1)
      ISIGN=1
1     DO 91 INN=N,NNN,N
      KNN=INN-N
      DO 12 J=1,N
      TEMPR=DATARL(J+KNN)
      TEMPI=DATAIM(J+KNN)
      DATARL(J+KNN)=DATARL(NTRANS(J)+KNN)
      DATAIM(J+KNN)=DATAIM(NTRANS(J)+KNN)
      DATAIM(NTRANS(J)+KNN)=TEMPI
12    DATARL(NTRANS(J)+KNN)=TEMPR
      NSS=N/2
      DO 15 J=1,NSS
      L=2*J-1+KNN
      M=L+1
      TEMPR=DATARL(L)+DATARL(M)
      TEMPI=DATAIM(L)+DATAIM(M)
      DATARL(M)=DATARL(L)-DATARL(M)
```

```
      DATAIM(M)=DATAIM(L)-DATAIM(M)
      DATAIM(L)=TEMPI
 15   DATARL(L)=TEMPR
      IF (N-2) 91,91,20
 20   DO 90 I=2,LOGN
      NUM=2**I
      NUMHF=NUM/2
      NSS=N/NUM
      DO 90 J=1,NSS
      NUMJK=NUM*(J-1)+KNN
      L=1+NUMJK
      M=L+NUMHF
      TEMPR=DATARL(L)+DATARL(M)
      TEMPI=DATAIM(L)+DATAIM(M)
      DATARL(M)=DATARL(L)-DATARL(M)
      DATAIM(M)=DATAIM(L)-DATAIM(M)
      DATARL(L)=TEMPR
      DATAIM(L)=TEMPI
      DO 90 K=2,NUMHF
      L=K+NUMJK
      M=L+NUMHF
      MM=NSS*(K-1)
      W2=W(2,MM)
      IF (ISIGN.GT.0) GO TO 80
      W2=-W2
 80   CROSSR=DATARL(M)*W(1,MM)+DATAIM(M)*W2
      CROSSI=DATAIM(M)*W(1,MM)-DATARL(M)*W2
      DATARL(M)=DATARL(L)-CROSSR
      DATAIM(M)=DATAIM(L)-CROSSI
      DATARL(L)=DATARL(L)+CROSSR
      DATAIM(L)=DATAIM(L)+CROSSI
 90   CONTINUE
 91   CONTINUE
      IF (ISIGN.LT.0) GO TO 99
      DO 97 I=1,NNN
      DATARL(I)=DATARL(I)/N
 97   DATAIM(I)=DATAIM(I)/N
 99   RETURN
      ENTRY FFTREV
      ISIGN=-1
      GO TO 1
      END
```

Routine FFTFOR performs the Fast Fourier Transform as a step in the transformation between the physical space variables and their spectral coefficients. First, the branch to statement 80 can be avoided if the variable W2 is properly set upon entry. This is very little pre-processing but would allow having a branch-free code sequence. Another branch after statement 15 can be removed as it serves no real purpose in practice.

Besides the pre-process setting, the code can be divided into 2 parts. The first part consists of loop 12 only to rearrange the input sequence in support of the branch-free computation that follows as the second part. The code is constructed to transform, at one time, a single input sequence whose elements are in the memory consecutively. Since the length of the sequence N=16 is a small number in the program, there is very little point to vectorize the transform itself. A better way of utilizing the FMP is to perform the transform in the same manner over many input sequences together. To do this, the sequences must be aligned in such a way that all the first elements of input sequences are consecutive in the memory, followed by all the second elements, etc. This involves a transpose operation which is time consuming in that one can expect as few as one single word/result per cycle.

```
      SUBROUTINE VFFTFOR(DATARL,DATAIM)
      COMMON /FFT/ . . .
      COMMON /FTCST/ . . .
      DIMENSION DATARL( . . .

      DIMENSION RL(NN,1),YM(NN,1)
      EQUIVALENCE (RL,DATARL),(YM,DATAIM)
      DYNAMIC PR,PI,TR,TI
      DYNAMIC RLO,RLDO,YMO,YMOO

      ISIGN=0
      W2=W0
    9 CONTINUE
```

```
      CALL REARR

      DO 15 L=1,N,2
      M=L+1
      PR=RLD(L)+RLD(M)
      RLD(M)=RLD(L)-RLD(M)
      RLD(L)=PR
      PI=YMD(L)+YMD(M)
      YMD(M)=YMD(L)-YMD(M)
      YMD(L)=PI
15    CONTINUE
      NUM=2
      NSS=N*.5
      DO 89 I=2,LOGN
      NUMHF=NUM
      NUM=NUM*2
      NSS=NSS*.5
      NUMJK=-NUM
      DO 89 J=1.NSS
      NUMJK=NUMJK+NUM
      L=1+NUMJK
      M=L+NUMJK
      PR=RLD(L)+RLD(M)
      RLD(M)=RLD(L)-RLD(M)
      RLD(L)=PR
      PI=YMD(L)+YMD(M)
      YMD(M)=YMD(L)-YMD(M)
      YMD(L)=PI0
      MM=0
      DO 89 K=2,NUMHF
      MM=MM+NSS
      L=K+NUMJK
      M=L+NUMHF
      TI=YMD(M)*W(1,MM)-RLD(M)*W2(MM)
      TR=RLD(M)*W(1,MM)+YMD(M)*W2(MM)
      YMD(M)=YMD(L)-TI
      YMD(L)=YMD(L)+TI
      RLD(M)=RLD(L)-TR
      RLD(L)=RLD(L)+TR
89    CONTINUE
89    CONTINUE
      IF (ISIGN.NE.-1) RETURN
      RLD0=RLD0*FN
      YMD0=YMD0*FN
      RETURN
      ENTRY FFTREV
      ISIGN=-1
      W2=-W0
      GO TO 9

      ENTRY FFT0
      DO 110 I=1,N
      DEFINE (RLD(I),RL(1:NN,I))
      DEFINE (YMD(I),YM(1:NN,I))
110   CONTINUE
      DEFINE (RLD0,RL(1:NNN,1))
      DEFINE (YMD0,YM(1:NNN,1))
      FN=1./N
      DO 111 I=1,7
      W0(I)=W(2,I)
111   CONTINUE
      RETURN
      END
```

This routine is one that cannot be vectorized from the point of view of the algorithm. There are 496 flops in each transform. Assuming no instruction has to wait for the previous result, each transform will need 496 cycles to get out of the Scalar Unit. Each time an FFT subroutine is called, it has to perform (15*NLEV+1)/2 transforms, where NLEV is the number of levels involved.

NLEV is less than or equal to 25. If NLEV=25, for example, there will be 188 transforms and 93,248 flops altogether.

If all transforms are performed through the pipeline as coded above, the number of cycles needed for NN transforms is 126M+867 where M=(NN+4)/8.

For NN=188, 3891 cycles are required, which yields 1.5 gigaflops. This is for the computations only.

```
      SUBROUTINE SPCFOR(ASPEC,AGRID,AI,NVERT)
      REAL ARSP(30),AGRID,AI
      COMMON /FFT/ WP(7,7,15),W(2,7),NTRANS(16),NNN,NN,LR1,NLATHF,
     1    NCPAR(7),LOGN
      COMMON /CONSTS/ J1(2),LR,J2(10),NREAL,NZONE
      COMMON /CGBLK/ J4(86),NCOMP(12)
      COMMON /FTCST/ N,NLAT,J6,ARSP
      COMMON /GLOP/ P(7,7,15),WT(50),AR(50)
      DIMENSION ASPEC(1),AGRID(1),AI(1)
      DO 5 I=1,NNN
      AGRID(I)=0.
 5    AI(I)=0
      DO 30 K=1,NVERT
      M=(K-1)*NLAT*N
      MM=(K-1)*NREAL
      DO 30 J=1,NLATHF
```

```
      JJ=NLAT+1-J
      ODDR=0.
      EVENR=0.
      LJ0=M+(J-1)*N
      LJJ0=M+(JJ-1)*N
      DO 10 I=1,NZONE,2
10    EVENR=EVENR+ASPEC(MM+I)*P(1,I,J)
      IF (LJ0.NE.LJJ0) GO TO 11
      AGRID(LJ0+1)=EVENR
      GO TO 16
11    DO 15 I=2,NZONE,2
15    ODDR=ODDR+ASPEC(MM+I)*P(1,I,J)
      AGRID(LJ0+1)=EVENR+ODDR
      AGRID(LJJ0+1)=EVENR-ODDR
16    ICE=NZONE+1
      ICO=NZONE+3
      DO 30 L=2,LR1
      ODDR=0.
      ODDI=0.
      EVENR=0.
      EVENI=0.
      IEND=NCOMP(L)
      LJ=LJ0+L
      LJJ=LJJ0+L
      DO 20 I=1,IEND,2
      EVENR=EVENR+ASPEC(MM+ICE)*P(L,I,J)
      ICE=ICE+1
      EVENI=EVENI+ASPEC(MM+ICE)*P(L,I,J)
20    ICE=ICE+3
      IF (LJ.NE.LJJ) GO TO 21
      AGRID(LJJ)=EVENR
      AI(LJJ)=EVENI
      GO TO 26
21    DO 25 I=2,IEND,2
      ODDR=ODDR+ASPEC(MM+ICO)*P(L,I,J)
      ICO=ICO+1
      ODDI=ODDI+ASPEC(MM+ICO)*P(L,I,J)
25    ICO=ICO+3
      AGRID(LJ)=EVENR+ODDR
      AI(LJ)=EVENI+ODDI
      AGRID(LJJ)=EVENR-ODDR
      AI(LJJ)=EVENI-ODDI
      LLJ=LJ0+N+2-L
      AGRID(LLJ)=AGRID(LJ)
      AI(LLJ)=-AI(LJ)
26    LLJJ=LJJ0+N+2-L
      AGRID(LLJJ)=AGRID(LJJ)
      AI(LLJJ)=-AI(LJJ)
      IF (NCPAR(L).EQ.0) GO TO 30
      ICK=ICO
      ICO=ICE
      ICE=ICK
30    CONTINUE
      RETURN
      ENTRY FORSPC
      DO 80 I=1,NVERT
      M=(I-1)*NLAT*N
      MM=(I-1)*NREAL
      DO 70 J=1,NZONE
      R=0.
      DO 60 K=1,NLAT
```

```
      LL=M+(K-1)*N+1
 60   R=R+AGRID(LL)*WP(1,J,K)
 70   ASPEC(MM+J)=R
      IC=NZONE
      DO 80 J=2,LR1
      IEND=NCOMP(J)
      DO 80 JJ=1,IEND
      R=0.
      C=0.
      IC=IC+1
      DO 75 K=1,NLAT
      LL=M+(K-1)*N+J
      R=R+AGRID(LL)*WP(J,JJ,K)
 75   C=C+AI(LL)*WP(J,JJ,K)
      ASPEC(IC+MM)=R
      IC=IC+1
      ASPEC(IC+MM)=C
 80   CONTINUE
      RETURN
      END
```

There are two parts in this subroutine to perform the Legendre transform in both directions. SPCFOR transforms the spectral coefficients into Fourier coefficients and FORSPC transforms the Fourier coefficients into the spectral coefficients. They are really independent of each other in instructions, although they share the same data block.

The data is structured in such a way that the spectral coefficients of a given level are grouped together and followed by those of the next level, and so on. The Fourier coefficients of a sequence are grouped together and followed by those of another sequence. This is a good structure for most conventional non-vector computers to process data in vector fashion, mainly to maximize the data flow rate. In fact, this entire program was coded originally in this manner to achieve vector-like performance with non-vector computers.

For the FMP, it is wiser to structure the data another way, namely, to group coefficients of same index but for all different levels and latitudes. This way, every instruction in the transform can be performed on an array of elements that are of the same index, rather than one single element. The algorithm can then be kept virtually unchanged. As a matter of fact, a closer look at the code will reveal that the algorithm is basically scalar in nature and vectorization of it is almost self-destructive.

One of the fundamental operations here is the inner product between two vectors. One machine instruction on the FMP will do this but it is designed to serve long vectors. In fact, for a vector length of 4, nothing will be accomplished. In the routine SPCFOR, the vector length will range from 3 to 4; in FORSPC, the vector length is 15. In both parts, the inner products are in the innermost loop. To avoid the drawback, the program is recoded in such a way that the inner product is broken into fundamental arithmetic operations and the loop for it is turned inside out as follows:

```
      SUBROUTINE VSPCFOR(ASPEC,AGRID,AI,NVERT)
         •
         •
         •
      DO 30 K=1,NVERT
      MM=(K-1)*NREAL
      DO 30 J=1,NLATHF1
      L=1
      EVENR(K,J,L)=ASPEC(MM+1)*P(L,1,J)+ASPEC(MM+3)*P(L,3,J)
      ODDR(K,J,L)=ASPEC(MM+2)*P(L,2,J)+ASPEC(MM+4)*P(L,4,J)
10    EVENR(K,J,L)=EVENR(K,J,L)+ASPEC(MM+5)*P(1,5,J)
15    ODDR(K,J,L)=ODDR(K,J,L)+ASPEC(MM+6)*P(1,6,J)
      EVENR(K,J,L)=EVENR(K,J,L)+ASPEC(MM+7)*P(L,7,J)
      AGRID(K,J,L)=EVENR(K,J,L)+ODDR(K,J,L)
      AGRID(K,NLAT+1-J,L)=EVENR(K,J,L)-ODDR(K,J,L)
16    ICE=NZONE-11
      DO 30 L=2,LR1
      ICE=ICE+12
      EVENR(K,J,L)=ASPEC(MM+ICE)*P(L,1,J)+ASPEC(MM+ICE+4)*P(L,3,J)
```

```
      EVENI(K,J,L)=ASPEC(MM+ICE+1)*P(L,1,J)+ASPEC(MM+ICE+5)*P(L,3,J)
      ODDR(K,J,L)=ASPEC(MM+ICE+2)*P(L,2,J)+ASPEC(MM+ICE+6)*P(L,4,J)
      ODDI(K,J,L)=ASPEC(MM+ICE+3)*P(L,2,J)+ASPEC(MM+ICE+7)*P(L,4,J)
      EVENR(K,J,L)=EVENR(K,J,L)+ASPEC(MM+ICE+8)*P(L,5,J)
      EVENI(K,J,L)=EVENI(K,J,L)+ASPEC(MM+ICE+9)*P(L,5,J)
20    CONTINUE
21    CONTINUE
      ODDR(K,J,L)=ODDR(K,J,L)+ASPEC(MM+ICE+10)*P(L,6,J)
      ODDI(K,J,L)=ODDI(K,J,L)+ASPEC(MM+ICE+11)*P(L,6,J)
25    CONTINUE
      AGRID(K,J,L)=EVENR(K,J,L)+ODDR(K,J,L)
      AI(K,J,L)=EVENI(K,J,L)+ODDI(K,J,L)
      AGRID(K,NLAT+1-J,L)=EVENR(K,J,L)-ODDR(K,J,L)
      AI(K,NLAT+1-J,L)=EVENI(K,J,L)-ODDI(K,J,L)
      AGRID(K,J,N+2-L)=AGRID(K,J,L)
      AI(K,J,N+2-L)=-AI(K,J,L)
26    CONTINUE
      AGRID(K,NLAT+1-J,N+2-L)=AGRID(K,NLAT+1-J,L)
      AI(K,NLAT+1-J,N+2-L)=-AI(K,NLAT+1-J,L)
30    CONTINUE

      DO 80 K=1,NVERT
      MM=(K-1)*NREAL
      L=1
      EVENR(K,NLATHF,L)=ASPEC(MM+1)*P(L,1,NLATHF)+ASPEC(MM+3)
     1     *P(L,3,NLATHF)
60    EVENR(K,NLATHF,L)=EVENR(K,NLATHF,L)+ASPEC(MM+5)*P(1,5,NLATHF)
      AGRID(K,NLATHF,L)=EVENR(K,NLATHF,L)+ASPEC(MM+7)*P(L,7,NLATHF)
66    ICE=NZONE-11
      DO 80 L=2,LR1
      ICE=ICE+12
      EVENR(K,NLATHF,L)=ASPEC(MM+ICE)*P(L,1,NLATHF)+ASPEC(MM+ICE+4)
     1     *P(L,3,NLATHF)
      EVENI(K,NLATHF,L)=ASPEC(MM+ICE+1)*P(L,2,NLATHF)+ASPEC(MM+ICE+5)
     1     *P(L,4,NLATHF)
70    CONTINUE
      AGRID(K,NLATHF,L)=EVENR(K,NLATHF,L)+ASPEC(MM+ICE+8)*P(L,5,NLATHF)
      AI(K,NLATHF,L)=EVENI(K,NLATHF,L)+ASPEC(MM+ICE+9)*P(L,5,NLATHF)
76    CONTINUE
      AGRID(K,NLATHF,N+2-L)=AGRID(K,NLATHF,L)
      AI(K,NLATHF,N+2-L)=-AI(K,NLATHF,L)
80    CONTINUE
      RETURN

      ENTRY FORSPC
      DO 180 I=1,NVERT
      MM=(I-1)*NREAL
      J=1
      DO 170 JJ=1,NZONE
      DO 160 K=1,NLATHF1
      KK=16-K
      R(I,JJ,J,K)=AGRID(I,K,J)*WP(J,JJ,K)+AGRID(I,KK,J)*WP(J,JJ,KK)
160   CONTINUE
      R(I,JJ,J,1)=AGRID(I,8,J)*WP(J,JJ,8)+R(I,JJ,J,1)
      R(I,JJ,J,3)=R(I,JJ,J,2)+R(I,JJ,J,3)
      R(I,JJ,J,5)=R(I,JJ,J,4)+R(I,JJ,J,5)
      R(I,JJ,J,7)=R(I,JJ,J,6)+R(I,JJ,J,7)
      R(I,JJ,J,1)=R(I,JJ,J,7)+R(I,JJ,J,1)
      R(I,JJ,J,3)=R(I,JJ,J,5)+R(I,JJ,J,3)
      ASPEC(MM+JJ)=R(I,JJ,J,3)+R(I,JJ,J,1)
170   CONTINUE

      IC=NZONE+1
      DO 180 J=2,LR1
      DO 180 JJ=1,IEND
```

```
      DO 174 K=1,NLATHF1
      KK=16-K
      R(I,JJ,J,K)=AGRID(I,K,J)*WP(J,JJ,K)+AGRID(I,KK,J)*WP(J,JJ,KK)
      C(I,JJ,J,K)=AI(I,K,J)*WP(J,JJ,K)+AI(I,KK,J)*WP(J,JJ,KK)
174   CONTINUE
      R(I,JJ,J,7)=R(I,JJ,J,7)+AGRID(I,8,J)*WP(J,JJ,8)
      C(I,JJ,J,7)=C(I,JJ,J,7)+AI(I,8,J)*WP(J,JJ,8)
      DO 175 K=1,5,2
      R(I,JJ,J,K)=R(I,JJ,J,K)+R(I,JJ,J,K+1)
      C(I,JJ,J,K)=C(I,JJ,J,K)+C(I,JJ,J,K+1)
175   CONTINUE
      DO 176 K=1,5,4
      R(I,JJ,J,K)=R(I,JJ,J,K)+R(I,JJ,J,K+2)
      C(I,JJ,J,K)=C(I,JJ,J,K)+C(I,JJ,J,K+2)
176   CONTINUE
      ASPEC(MM+IC)=R(I,JJ,J,1)+R(I,JJ,J,5)
      ASPEC(MM+IC+1)=C(I,JJ,J,1)+C(I,JJ,J,5)
      IC=IC+2
180   CONTINUE
      RETURN

      ENTRY INT
      NLATHF1=NLATHF-1
      IEND=NCOMP(2)
      RETURN
      END
```

The intermediates R, C, EVENR, EVENI, ODDR, and ODDI have been made vectors, and the variables AGRID and AI have been re-dimensioned. The new dimensional structure is compatible with the counterpart in the routine FFTFOR/FFTREV. However, the structure of the array ASPEC is for this moment left untouched. In fact, what has been done in the above is far from vectorization intentionally because the purpose was to retain the main code structure so that it is easily followed. Though it is not at all in vector form, it can be put in vector form in a straightforward manner for one to see without having it actually carried out. The key here is to move the loop over the level into the innermost. The dimensional structure reflects the order of the loops, and therefore the form of the vectors.

SPCFOR is broken into two parts to avoid branches, leaving a branch-free code!!

The performance is incredibly good; nearly the full speed of the FMP can be realized as much of the instructions are 3-operation combinations. The only consideration remaining is to restructure ASPEC before SPCFOR and after FORSPC. This is slow, but a proper segmentation can be arranged so that the computation would be done for free.

The index of ASPEC takes the form of MM + K where MM provides the starting point of the level L, and K gives the index of the spectral coefficient. Conceivably, the data of the entire program could have been restructured with ASPEC(L,K) in place of ASPEC(MM+K). In that case, no further restructuring such as pre- or post-processing is required.

Most subroutines of the original code not discussed above were recoded using multiple indices, i.e., replacing Z(MM+K) with Z(K,L) in syntax. These are found is appendix D where each subroutine appears followed by the recoded version which has the same name except prefixed with a V. It can be noticed that one of the most common structures is as follows

```
DO 200 K=K1, K2           \
DO 100 L=L1, L2           |
Z(K,L) = ... X(K,L) ...    >  (22)
100 CONTINUE              |
200 CONTINUE              /
```

Loops which are essentially of the form (22) have been identified by brackets around them in the left margin of appendix D.

A double loop in this very form is not collapsible for vectorization because the elements of the array Z and X are not visited consecutively. However, if the data array is restructured in such a way that the indices K and L interchange their positions, the result is

```
DIMENSION Z(LL,KK), X(LL,KK)   \
DO 200 K=1, KK                 |
DO 100 L=1, LL                 |
                                >  (23)
Z(L,K) = ... X(L,K) ...        |
100 CONTINUE                   |
200 CONTINUE                   /
```

This same double loop is then readily reducible to one single vector instruction of length LL*KK.

Simple loops are identified by an arrow in the left margin of appendix D; these are, of course, readily vectorizable. Other loops which are not quite in the form of (22) are marked with brackets in the right margin of appendix D. However, these can be converted to the form of (22) by merely interchanging the order of the double loop; these, in turn, can be changed to the vectorizable form of (23) by restructuring of data.

What this amounts to is essentially that each data array not only can be, but also should be, restructured into its transpose. In other words, elements of same index (spectral or physical) but of different levels should be grouped together, followed by a group of another index, etc. This will virtually render the entire program instantly vectorizable without change to the program logic. In this sense, the code in its original form is suitable for the FMP.

Some comments are in order regarding the streamlining of the code for high performance. Although these considerations have to be treated on an individual basis, they are by no means obscure and should be rather obvious to a reasonably well-trained programmer.

It can be noted that level delimiters, say L1 and L2, frequently appear in the code such that most loops run from L1 to L2. Data should be structured so that the loop can be transformed into a single long vector from L1 to L2. Elements for L < L1 and L > L2 can be grouped separately and processed differently. Conceivably, these are few and could be processed by the Scalar Unit in parallel with the vector operations.

The 79 components of the spectral coefficients for a 4-level case are originally ordered as shown in figure 14. Elements of the same index but different level should be grouped together and, in addition, the data should be structured in three portions as shown in figures 15a, 15b, and 15c. This avoids the necessity to Compress in VCORFOR and Compress/Merge in VSPCFOR, ... , etc.

In the physical domain, the data block should be structured as A(LONG1:LONG2, LAT1:LAT2, LVL1:LVL2). This change of the data

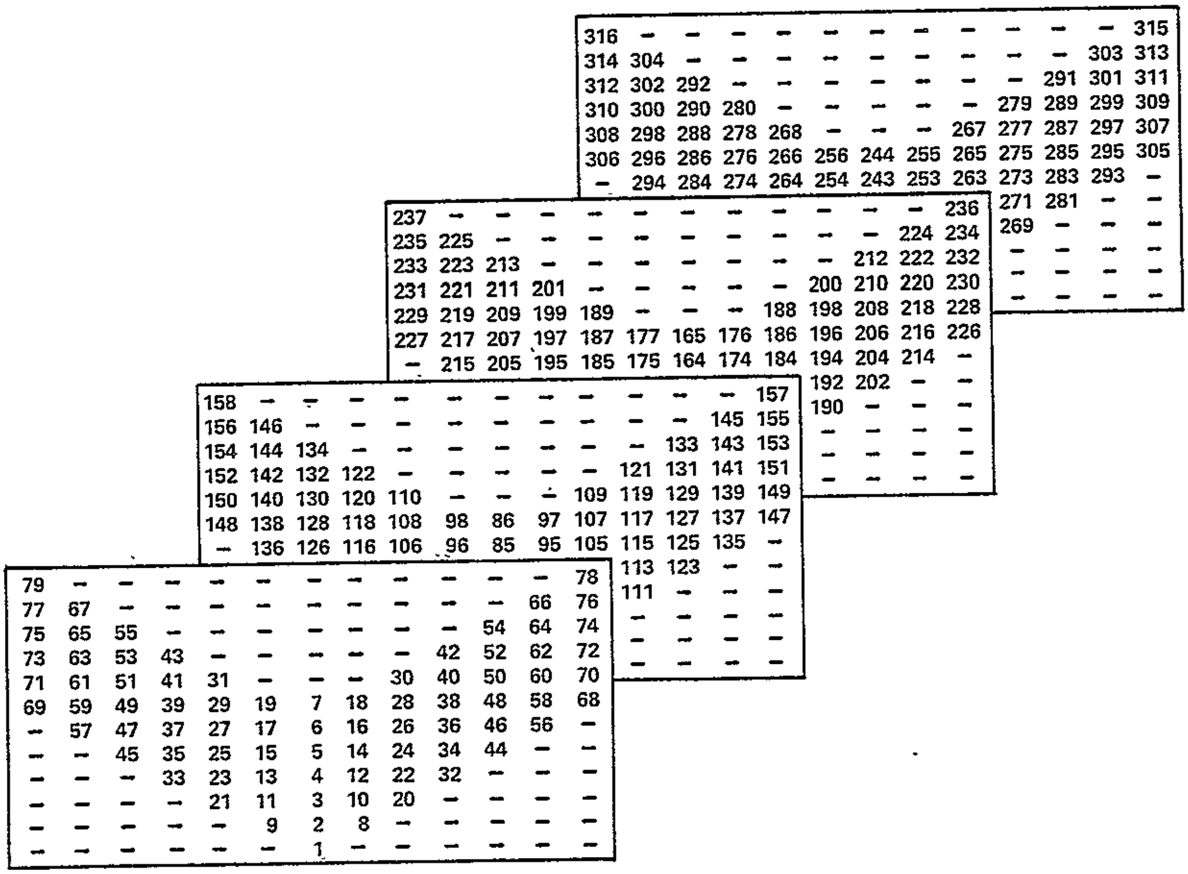
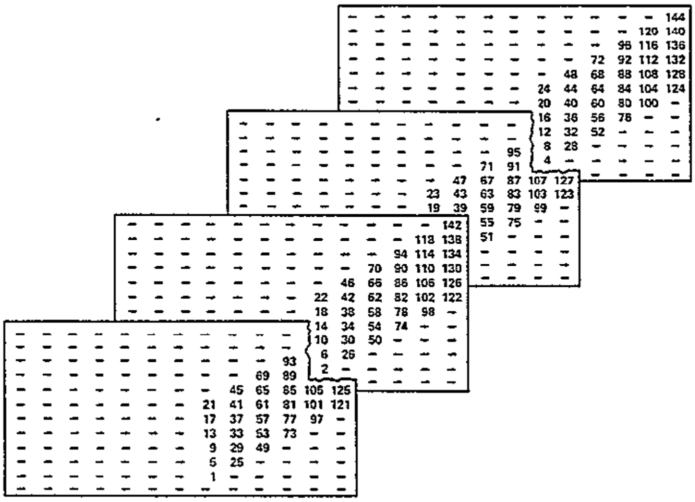
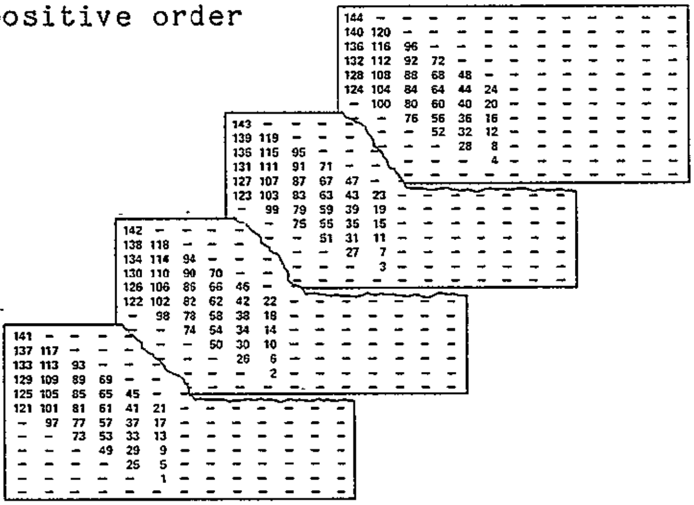


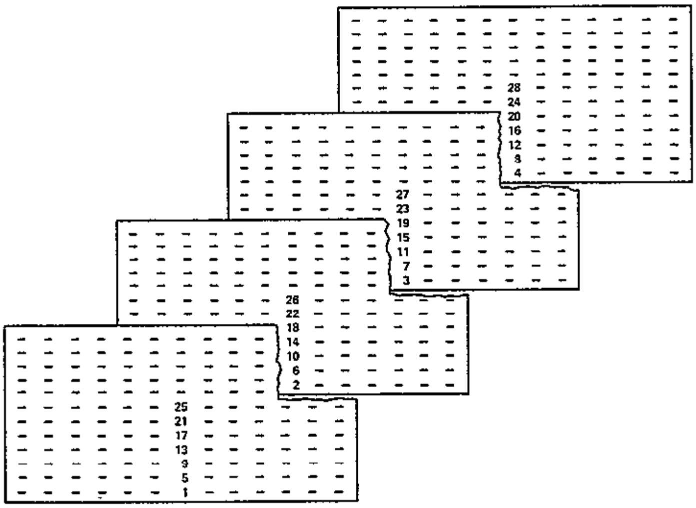
Figure 14. Ordering of Spectral Components in the Original Code/Data for a 4-Level Case as an Example.



a) Components of positive order



b) Components of negative order



c) Components of zeroth order

Figure 15. A Restructure and Reordering of the Data Block for Spectral Components in Figure 14.

structure will not effect the above analysis for O3CHEM, RTECON, and CHEMEQ if the code is integrated so as to remove the subroutine calls.

With the restructuring of data just discussed, the transforms SPCFOR and FFTFOR are free from the time-consuming task of data shuffling. As a matter of fact, REARR can be removed from VFFTFOR if loop 110 under ENTRY FFTO is changed:

```
     DO 110 I=1,N
     ITEMP = NTRANS(I)
     DEFINE (RLD(I), RL(1:NN, ITEMP))
     DEFINE (YMD(I), YM(1:NN, ITEMP))
110 CONTINUE
```

Total analysis and evaluation of the spectral code was not completed; time and resource limitations precluded carrying it to a point where it could be run on a benchmark basis. Relative importance in execution time of each routine is not obvious. An educated estimate of performance can be extrapolated from the analysis which was completed as being one gigaflop or better for the spectral code.

APPENDIX A

AVRX ROUTINE FROM GISS MODEL

```
      SUBROUTINE AVRX( PU )
C
      DIMENSION PU(416),NM(16),ALPHA(16),X(26),Y(26)
      DATA NM/0,3,1,1,1,0,0,0,0,0,0,1,1,1,3,0/
      DATA ALPHA/0.,1.186572E-1,1.208591E-1,4.513013E-2,9.563327E-3,
     * 0.,0.,0.,0.,0.,0.,9.563327E-3,4.513013E-2,1.208591E-1,
     * 1.186572E-1,0./
C
C   SMOOTHES THE ZONAL MASS FLUX AND GEOPOTENTIAL GRADIENTS
C   NEAR THE POLES TO HELP AVOID COMPUTATIONAL INSTABILITY
C
C   NOTE. THIS ROUTINE HAS BEEN SLIGHTLY ALTERED
C
      DO 40 J=2,15
      IF (NM(J).LE.0) GO TO 40
      J1=26*(J-1)+1
      J2=JI+1
      NMJ=NM(J)
      DO 30 N=1,NMJ
      X(2:24) = PU(J2:24) - PU(J1:24)
      X(1)=X(25)
      X(26)=X(2)
      Y(1:25) = X(2:25) - X(1:25)
      Y(1:25) = Y(1:25) * ALPHA(J)
      PU(J1:25) = PU(J1:25) + Y(1:25)
   30 CONTINUE
   40 CONTINUE
      RETURN
      END
```

APPENDIX B

LINKHO ROUTINE FROM GISS MODEL

```
      SUBROUTINE LINKHO                                                   0007820LINKHO
      INTEGER LIST(10)                                                    0007821LINKHO
      DATA LIST(1)/1/,NLIST/1/                                            0007822LINKHO
      INTEGER I1(384)                                                     0007823LINKHO
      REAL RI1(384)                                                       0007824LINKHO
      COMMON /EINT/TE3                                                    0007825LINKHO
C**** FOR COARSE GRID IM=24 AND JM=16                                     0007826LINKHO
C**** 384=IM*JM                                                           0007827LINKHO
C**** 768=2*IM*JM                                                         0007828LINKHO
C**** 1152=3*IM*JM                                                        0007829LINKHO
C**** 1536=4*IM*JM                                                        0007830LINKHO
C**** 1920=5*IM*JM                                                        0007831LINKHO
C**** 2304=6*IM*JM                                                        0007832LINKHO
C**** 2688=7*IM*JM                                                        0007833LINKHO
C**** 3072=8*IM*JM                                                        0007834LINKHO
C**** 3456=9*IM*JM                                                        0007835LINKHO
C**** 3840=10*IM*JM                                                       0007836LINKHO
C**** 4224=11*IM*JM                                                       0007837LINKHO
C**** 4608=12*IM*JM                                                       0007838LINKHO
C**** INTERFACE                                                           0007839LINKHO
      COMMON /MAICOM/JM,IM,NLAY,PTROP,JMM1,FIM,NLAYM1,NLAYP1,DLAT,DLON,   0007840LINKHO
     *  ISTART,KM,XTAUTX,IROT,MRUT,                                       0007841LINKHO
     *DT,XTAUX,ITAU,XINT,IDAY,JDAY,TOFDAY,JDATE,JMONTH(2),JYEAR,NSTEP,    0007842LINKHO
     *  NCYCLE,NCOMP3,NMOGAN,TAUP,TAUI,TAUE,TAUO,MRCH,                    0007843LINKHO
     *  PI,GRAV,RGAS,KAPA,PSL,ED,FMU,NFLW,PSF,RSDIST,SIND,COSD,RHMAX,     0007844LINKHO
     *  CDX,DUMMYC(151),                                                  0007845LINKHO
     *  XLABEL(20),SIG(20),DSIG(20),SIGE(21),DSIGO(19),                   0007846LINKHO
     *  LAT(16),SINL(16),COSL(16),DXU(16),DXP(16),DYU(16),DYP(16),        0007847LINKHO
     *  DXYP(16),F(16),DUMMY(24),                                         0007848LINKHO
     *              TS(384),         SHS(384),         GT(384),           0007849LINKHO
     *              GW(384),         PHIS(384),        TOPOG(384),        0007850LINKHO
     *        U(3456),V(3456),T(3456),SH(3456)                            0007851LINKHO
      COMMON/RADCOM/ PL(3456),PLE(3840),PKK(3456),TG(384),                0007852LINKHO
     * TSTR(1152),CLOUD(4608),RE(3840),RESTR(1152),                       0007853LINKHO
     * FLXDNG(384),SG(384),AS(3456),ASSTR(1152),SOX(384),COSZ(384),       0007854LINKHO
     * RSURF(384),SCOSZ(384),RAP(384),RAM(384),PLK(3456)                  0007855LINKHO
      COMMON/CLDCOM/SWALE(3840),SWIL(3456),AL(3840),TAUL(3840),           0007856LINKHO
     * OZALE(6144),TOPABS(384)                                            0007857LINKHO
      COMMON /TMPBLK/ A,AA,AAA,BB,CC,SE,BE,UN1,TAU1,TAUT,TNSQ,TN,         0007858LINKHO
     .   TAU,AERU,AERV,AERC,EX1,EX2,DENO,DNMO,DNM1,PEFUP,PEFDN,           0007859LINKHO
     .   PIOC,TAUNC,BTOPN,BTOPNP,EUPCN,                                   0007860LINKHO
     .   RUPCN,X1,X2,X3,RI1                                               0007861LINKHO
      COMMON /LNK/ EUP,RDNC,REF,TDFC,TDF,EDNC,EUPC,EDN,TAUN,FE,BTOP,TE,   0007862LINKHO
     .   E,P,UNO3,UNCO2,UNH2O,NCLOUD                                      0007863LINKHO
      DIMENSION  TL(3456), SHL(3456)                                      0007864LINKHO
      EQUIVALENCE (TL(1),T(1)), (SHL(1),SH(1))                            0007865LINKHO
C****                                                                     0007866LINKHO
C**** GRID ARRAYS                                                         0007867LINKHO
      BIT CLDFLG(384),AERFLG(384),L1(384),L2(384),L3(384)                 0007868LINKHO
      BIT TSTEXP(384)                                                     0007869LINKHO
      DIMENSION ZERO(384),ONE(384)                                        0007870LINKHO
      INTEGER IL1(6),IL2(6),IL3(6)                                        0007871LINKHO
      INTEGER ICLD(6),IAER(6)                                             0007872LINKHO
      INTEGER ITY(384)                                                    0007873LINKHO
      REAL RITY(384)                                                      0007874LINKHO
      REAL A(384),AAA(384),SE(384),BE(384),UN1(384),                      0007875LINKHO
     * TAU1(384),TAUT(384),AA(384),BB(384),CC(384),TNSQ(384),             0007876LINKHO
     * TN(384),TAU(384),EDNCN(384),TDFCN(384),RDNCN(384),                 0007877LINKHO
     * TAUCIR(384),PIO(384),EXTAU(384),TY(384),AER1(384),                 0007878LINKHO
     * AER2(384),AERA(384),AERU(384),AERV(384),AERC(384),                 0007879LINKHO
     * EX1(384),EX2(384),DENO(384),DNMO(384),DNM1(384),                   0007880LINKHO
     * PEFUP(384),PEFDN(384)                                              0007881LINKHO
      REAL X1(384),X2(384)                                                0007882LINKHO
```

```
      REAL XX1(3072),XX2(3072),PLKE(3072),TH(3456)                      0007883LINKHO
      REAL EUPCN(384),RUPCN(384),X3(384)                                0007884LINKHO
      REAL PI0C(384),TAUNC(384),BTOPN(384),BTOPNP(384)                  0007885LINKHO
      DIMENSION UNH20(4608),UNC02(4608),UNO3(4608),P(4608),             0007886LINKHO
     * E(4608),NCLOUD(4608),TE(5376),BTOP(5376),FE(4992),               0007887LINKHO
     * TAUN(13824), FKGAS(1152),FKGAS2(1152),EUP(4608),                 0007888LINKHO
     * EDN(4608),EUPC(4608),EDNC(4608),TDF(4608),TDFC(4608),            0007889LINKHO
     * REF(4608),RDNC(4608)                                             0007890LINKHO
      EQUIVALENCE (ITY(1),RITY(1)),(I1(1),RI1(1))                       0007891LINKHO
      EQUIVALENCE (FKGAS(1),EUP(1)),(FKGAS2(1),EUP(1153))               0007892LINKHO
      EQUIVALENCE (XX1(1),TAUN(1)),(XX2(1),TAUN(4609)),                 0007893LINKHO
     * (PLKE(1),TAUN(9217)),(TH(1),EUP(1))                              0007894LINKHO
      EQUIVALENCE (A(1),EDNCN(1)),(AAA(1),TDFCN(1)),(SE(1),RDNCN(1)),   0007895LINKHO
     * (BE(1),TAUCIR(1)),(UN1(1),PI0(1)),(TAU1(1),EXTAU(1)),            0007896LINKHO
     * (TAUT(1),TY(1)),(AA(1),ITY(1)),(AER1(1),BB(1)),(AER2(1),CC(1)),  0007897LINKHO
     * (TNSQ(1),AERA(1))                                                0007898LINKHO
      EQUIVALENCE (L1(1),IL1(1)),(L2(1),IL2(1)),(L3(1),IL3(1))          0007899LINKHO
      EQUIVALENCE (CLDFLG(1),ICLD(1)),(AERFLG(1),IAER(1))               0007900LINKHO
C*****                                                                  0007901LINKHO
C*****SCALAR ARRAYS (TABLES OR USED FOR INITIALIZATION)                  0007902LINKHO
      DIMENSION CB(12,12),PIZ(12,12),TA(12,12),PF1(12),PF2(12),         0007903LINKHO
     * TEMP(23),TE3(301),DV(11),PIAERO(12,2),NAERO(12),                 0007904LINKHO
     * ACOSBR(12,2),AEREXT(12,2),ATAU55(2),PICIRO(12),                  0007905LINKHO
     * CIREXT(12),CCOSBR(12),COELAM(12),COEK(3)                         0007906LINKHO
      DIMENSION SH20(3,3),BH20(3,3),WK(5,3),A1(12,3),A2(12,3),A3(12,3), 0007907LINKHO
     *A4(12,3),B1(3,3,2),B2(3,3,2),B3(3,3,2),C1(2,3),C2(2,3),WK20(2,2)  0007908LINKHO
C*****DATA V/240,360,480,560,680,760,840,960,1050,1160,1320,1560/       0007909LINKHO
      DATA DV/2*60.,40.,60.,2*40.,60.,45.,55.,80.,120./                 0007910LINKHO
C*****20 MICRON WATER VAPOR CONTINUUM                                    0007911LINKHO
      DATA WK20/2.6518E-03,7.2321E-04,6.1875E-02,4.0982E-02/            0007912LINKHO
C*****CIRRUS CLOUD PROPERTIES                                            0007913LINKHO
      DATA CTAU55/1.E0/                                                 0007914LINKHO
      DATA CCOSBR/0.827220,0.812128,0.770656,0.787898,0.884853,0.906536,0007915LINKHO
     *   0.928219,0.936090,0.941993,0.937202,0.945603,0.963118/         0007916LINKHO
      DATA CIREXT/1.291097,1.431162,1.025916,0.861792,0.783619,0.763708,0007917LINKHO
     *   0.743796,0.770507,0.790540,0.742040,0.688896,0.643580/         0007918LINKHO
      DATA PICIRO/0.510881,0.747140,0.680009,0.524641,0.278540,0.249016,0007919LINKHO
     *   0.219492,0.349411,0.446850,0.353371,0.263841,0.161871/         0007920LINKHO
C*****AEROSOL PROPERTIES                                                 0007921LINKHO
      DATA ATAU55/2*0.0/                                                0007922LINKHO
      DATA NAERO/12*0/                                                  0007923LINKHO
      DATA ACOSBR/0.00331,0.00703,0.01093,0.01589,0.02011,0.00000,      0007924LINKHO
     *     0.03478,0.00000,0.04604,0.04875,0.05902,0.09297,             0007925LINKHO
     *                0.28075,0.33668,0.35990,0.46584,0.59244,0.00000,  0007926LINKHO
     *     0.55580,0.00000,0.39450,0.69517,0.71047,0.69187/             0007927LINKHO
      DATA PIAERO/0.00248,0.00678,0.00798,0.01180,0.01817,0.00000,      0007928LINKHO
     *     0.05839,0.00000,0.04052,0.03904,0.01792,0.09674,             0007929LINKHO
     *                0.00248,0.00678,0.00798,0.01180,0.01817,0.00000,  0007930LINKHO
     *     0.05839,0.00000,0.04052,0.03904,0.01792,0.09674/             0007931LINKHO
      DATA AEREXT/0.00637,0.00968,0.01900,0.03454,0.02274,0.00000,      0007932LINKHO
     *     0.02652,0.00000,0.13152,0.18389,0.10690,0.04951,             0007933LINKHO
     *                0.03343,0.05528,0.08571,0.07775,0.05821,0.00000,  0007934LINKHO
     *     0.07908,0.00000,0.15930,0.08634,0.06592,0.07536/             0007935LINKHO
C*****PLANCK FUNCTION COEFFICIENT AT V                                   0007936LINKHO
      DATA PF1/1.06671,3.60014,8.53366,13.5511,24.2626,33.8729,         0007937LINKHO
     *          45.7351,68.2692,89.3263,120.444,177.473,292.944/        0007938LINKHO
      DATA PF2/345.319,517.979,690.638,805.745,978.404,1093.51,         0007939LINKHO
     *          1208.62,1381.28,1510.77,1669.04,1899.26,2244.57/        0007940LINKHO
C*****QUADRATURE FIT COEFFICIENTS OF S,B  OF WATER VAPOR AT 680,760,10500007941LINKHO
      DATA SH20/6.8618E-03,3.3054E-03,1.4435E-04,-1.9453E-02,-8.9078E-030007942LINKHO
     * ,-3.9394E-04,1.4498E-02,6.1017E-03,2.7344E-04/                   0007943LINKHO
      DATA BH20/-1.0739E-02,1.2100E-01,9.6612E-03,5.8873E-02,-0.2536E 000007944LINKHO
     *,2.3355E-02,-3.2402E-02,1.5054E-01,-1.0194E-02/                   0007945LINKHO
```

```
C*****QUADRATURE FIT COEFFICIENTS OF WATER VAPOR CONTINUUM AT 840,1050, 0007946LINKHO
C*****AND 1160 CM-1                                                      0007947LINKHO
      DATA WK/0.2227614,0.1013448,0.0828664,0.060817,0.057787,-0.30915280007948LINKHO
     *,-0.133841,-0.1138839,-0.080575,-0.077208,0.1093719,0.045289,      0007949LINKHO
     *0.039877,0.027158,0.026355/                                        0007950LINKHO
C*****QUADRATURE FIT COEFFICIENTS OF WATER VAPOR SELECTIVE ABSORP        0007951LINKHO
      DATA A1/5.2200E-02,3.7670E-01,6.7870E-01,5.6300E-02,1.6690E-02,    0007952LINKHO
     *         4.7784E-02,2.9330E-03,0.0000E 00,                         0007953LINKHO
     1                       7.9000E-04,3.4790E-03,5.0278E-01,3.1559E 01, 0007954LINKHO
     2         1.8971E 01,-6.390E 00,-1.880E 00,-1.930E-01,-5.131E-02,   0007955LINKHO
     *         -7.4814E-02,-7.795E-03,0.0000E 00,                        0007956LINKHO
     3                       -2.156E-03,-1.044E-02,-1.541E 00,-3.3418E 01, 0007957LINKHO
     4         9.5713E 00,1.0424E 01,1.3858E 00,1.9100E-01,4.1490E-02,   0007958LINKHO
     *         2.9366E-02,5.2520E-03,0.0000E 00,                         0007959LINKHO
     5                       1.4930E-03,8.2320E-03,1.3180E 00,1.4540E 01/ 0007960LINKHO
      DATA A2/1.9924,1.3814,-0.6081,1.0755,-0.6334,8.5722,-0.0990,0.00000007961LINKHO
     *,0.2136, 0.8515,15.0347,3.7794,-1.6729,1.2349,2.3437,6.211E-03,    0007962LINKHO
     *3.271,-14.4357,1.854,0.0000,2.0357,0.1680,-21.886,-2.0612,1.0139,  0007963LINKHO
     *-1.1483,-0.4787,-3.105E-03,-1.8006,6.8275,-0.7560,0.0000,-0.9935,  0007964LINKHO
     *8.722E-02,9.9179,1.0342/                                           0007965LINKHO
      DATA A3/3*2.674E-03,2.685E-03,2.695E-03,2.677E-03,2.687E-03,       0007966LINKHO
     *0.000E 00,2.686E-03,2.684E-03,2.605E-03,2.614E-03,3*1.875E-02,     0007967LINKHO
     *1.883E-02,1.890E-02,1.878E-02,1.884E-02,0.000E 00,1.880E-02,       0007968LINKHO
     *1.881E-02,1.826E-02,1.833E-02,12*1.0/                              0007969LINKHO
      DATA A4/0.02042,0.02050,0.02051,0.02114,0.02246,0.02061,0.02149,   0007970LINKHO
     *0.0000,0.01716,0.02114,0.01798,0.01825,2*0.0866,0.0868,0.0887,     0007971LINKHO
     *0.09089,0.08739,0.08918,0.0000,0.0876,0.0881,0.0791,0.07993,       0007972LINKHO
     *12*1.0/                                                            0007973LINKHO
C*****QUADRATURE FIT COEFFICIENT OF CARBON DIOXIDE AT 680  760 CM-1       0007974LINKHO
      DATA B1/1.2797,6.8537,205.7428,-1.8824,-11.8905,-277.6561,1.6415,  0007975LINKHO
     110.0278,294.3491,                                                  0007976LINKHO
     *1.3940E-02,9.7310E-02,4.0701E 00,-2.0577E-02,-1.4362E-01,          0007977LINKHO
     *-5.9580E 00,7.6840E-03,5.3619E-02,2.2095E 00/                      0007978LINKHO
      DATA B2/0.8390,0.7830,6.3778,-0.03698,-0.7746,-3.2630,-0.03424,    0007979LINKHO
     11.0889,12.1420,                                                    0007980LINKHO
     *4.2260E-02,1.2300E-01,8.4814E 01,1.8314E-01,9.6351E-01,-1.2326E 020007981LINKHO
     *,-6.6549E-02,-2.9198E-01,5.0527E 01/                               0007982LINKHO
      DATA B3/0.1639,0.9301,1.3578,-0.2338,0.06693,-0.3961,0.2574,       0007983LINKHO
     1-0.02676,0.2506,                                                   0007984LINKHO
     *-4.9551E-01,-5.2184E-02,1.3029E-01,1.0379E 00,1.1386E-01,          0007985LINKHO
     *-1.6563E-01,3.7517E-01,9.2239E-01,1.2554E 00/                      0007986LINKHO
C*****QUADRATURE FIT COEFFICIENT OF OZONE AT 1050 CM-1                    0007987LINKHO
      DATA C1/14.1003,89.8407,-14.3659,-81.9237,9.7780,59.5051/          0007988LINKHO
      DATA C2/1.0952,2.9766,1.3823,11.9258,-0.1894,-3.2240/              0007989LINKHO
      DATA IDATA/0/                                                      0007990LINKHO
C     FTEMP(X,Y,Z,T)=X+Y*T+Z*T*T                                         0007991LINKHO
C*****                                                                   0007992LINKHO
C*****9 LAYERS DYNAMICAL MODEL IS USED WITH 3   EXTRA LAYER AT 0-10 MB    0007993LINKHO
C*****                                                                   0007994LINKHO
      IF(IDATA) 2000,2000,2001                                           0007995LINKHO
2000  CONTINUE                                                           0007996LINKHO
      NLAY=9                                                             0007997LINKHO
C*****GROUND ALBEDO                                                      0007998LINKHO
      AGRND=0.                                                           0007999LINKHO
      NLAY1=NLAY+1                                                       0008000LINKHO
      NLAYRS=NLAY+3                                                      0008001LINKHO
      NG=NLAYRS+1                                                        0008002LINKHO
      NG1=NG+1                                                           0008003LINKHO
C*****CM-STP/MB    5.11E-4*2.24E4/44/G                                   0008004LINKHO
      CO2CM=2.65287E-01                                                  0008005LINKHO
C*****STRATOSPHERIC WATER VAPOR MIXING RATIO                             0008006LINKHO
      H2OMIX=3.E-06                                                      0008007LINKHO
      COEK(1)=.293478                                                    0008008LINKHO
```

```
      COEK(2)=.413043                                                   0008009LINKHO 191
      COEK(3)=COEK(1)                                                   0008010LINKHO 192
      COELAM(1)=DV(1)+120.                                              0008011LINKHO 193
      COELAM(12)=DV(11)+400.                                            0008012LINKHO 194
      DO 43 LAM=2,11                                                    0008013LINKHO 195
   43 COELAM(LAM)=DV(LAM)+DV(LAM-1)                                     0008014LINKHO 196
C**** CALCULATE COSBAR AND TAUAER*PIAERO                                0008015LINKHO 197
      DO 51 N=1,NLAYRS                                                  0008016LINKHO 198
      DO 51 LAM=1,12                                                    0008017LINKHO 199
      GO TO (52,52,52,53,52,52,52,52,52,52,52,54),N                     0008018LINKHO 200
53    TA(LAM,N)=AEREXT(LAM,1)*ATAUS5(1)                                 0008019LINKHO 201
      NN=1                                                              0008020LINKHO 202
      GO TO 58                                                          0008021LINKHO 203
54    TA(LAM,N)=AEREXT(LAM,2)*ATAUSS(2)                                 0008022LINKHO 204
      NN=2                                                              0008023LINKHO 205
      GO TO 58                                                          0008024LINKHO 206
52    TA(LAM,N)=0.                                                      0008025LINKHO 207
C***************************************************************        0008026LINKHO 208
C NN UNDEFINED FOR THIS BRANCH IN ORIGINAL                              0008027LINKHO 209
C SET IT TO 1 TO PREVENT MACHINE PROBLEMS                               0008028LINKHO 210
C***************************************************************        0008029LINKHO 211
      NN=1                                                              0008030LINKHO 212
58    CB(LAM,N)=(ACOSBR(LAM,NN)*AEREXT(LAM,NN)+CCOSBR(LAM)*CIREXT(N))/  0008031LINKHO 213
     * (AEREXT(LAM,NN)+CIREXT(N)+1.E-40)                                0008032LINKHO 214
      PIZ(LAM,N)=TA(LAM,N)*PIAERO(LAM,NN)                               0008033LINKHO 215
51    CONTINUE                                                          0008034LINKHO 216
      H2OFAC=H2OMIX/(H2OMIX+1.)*1.27E3                                  0008035LINKHO 217
      ZERO(1;384) = 0.0                                                 0008036LINKHO 218
      ONE(1;384) = 1.0                                                  0008037LINKHO 219
C.....FILL UP TSTR FIRST TIME AROUND ONLY                               0008038LINKHO 220
      TSTR(1;1152) = 200.0                                              0008039LINKHO 221
      IDATA=1                                                           0008040LINKHO 222
2001  CONTINUE                                                          0008041LINKHO 223
C****                                                                   0008042LINKHO 224
C**** CHANGE GRID TO 12 LAYERS                                          0008043LINKHO 225
C**** COMPUTE LAYER THICKNESS--STORED IN UNCO2                          0008044LINKHO 226
C**** UNCO2(1)=PE(2)-PE(1)=2.-0.=2.                                     0008045LINKHO 227
C**** UNCO2(2)=PE(3)-PE(2)=5.-2.=3.                                     0008046LINKHO 228
C**** UNCO2(3)=PE(4)-PE(3)=10.-5.=5.                                    0008047LINKHO 229
      UNCO2(1;384)=2.                                                   0008048LINKHO 230
      UNCO2(385;384)=3.                                                 0008049LINKHO 231
      UNCO2(769;384)=5.                                                 0008050LINKHO 232
C UNCO2(N)=PLE(N-2)-PLE(N-3) ..., N=4,NLAYRS                            0008051LINKHO 233
      UNCO2(1153;3456)=PLE(385;3456)-PLE(1;3456)                        0008052LINKHO 234
C**** COMPUTE H2O PARTIAL PRESSURE                                      0008053LINKHO 235
C FOR LAYERS 1 TO 3 UNH2O=H2OMIX/(1+H2OMIX)*1.27E3*(LAYER THICKNESS)    0008054LINKHO 236
C FOR LAYERS 4 TO 12 UNH2O=(SHL(N-3)+1.E-20)*1.27E3*(LAYER THICKNESS)   0008055LINKHO 237
      UNH2O(1;1152)=H2OFAC*UNCO2(1;1152)                                0008056LINKHO 238
      UNH2O(1153;3456)=SHL(1;3456)+1.E-20                               0008057LINKHO 239
      UNH2O(1153;3456)=1.27E3*UNH2O(1153;3456)                          0008058LINKHO 240
      UNH2O(1153;3456)=UNH2O(1153;3456)*UNCO2(1153;3456)                0008059LINKHO 241
C**** COMPUTE OZONE PARTIAL PRESSURE                                    0008060LINKHO 242
      UNO3(1;384)=OZALE(4225;384)-OZALE(3841;384)                       0008061LINKHO 243
      UNO3(385;384)=OZALE(4609;384)-OZALE(4225;384)                     0008062LINKHO 244
      UNO3(769;384)=OZALE(1;384)-OZALE(4609;384)                        0008063LINKHO 245
      UNO3(1153;3456)=OZALE(385;3456)-OZALE(1;3456)                     0008064LINKHO 246
C**** COMPUTE NORMALIZED PRESSURE                                       0008065LINKHO 247
C     PE(1)=0.                                                          0008066LINKHO 248
C     PE(2)=2.                                                          0008067LINKHO 249
C     PE(3)=5.                                                          0008068LINKHO 250
C     PE(4)=10.                                                         0008069LINKHO 251
C     DO 48 N=1,3                                                       0008070LINKHO 252
C     PSTR(N)=(PE(N)+PE(N+1))/2.                                        0008071LINKHO 253
```

```
C  48 P(N)=PSTR(N)/1013.25                                              0008072LINKHO 254
      P(1;384)=.98692323E-3                                             0008073LINKHO 255
      P(385;384)=.34542314E-2                                           0008074LINKHO 256
      P(769;384)=.74019209E-2                                           0008075LINKHO 257
      P(1153;3456)=PL(1;3456)/1013.                                     0008076LINKHO 258
C**** COMPUTE E (H2O ABSORPTION PARAMETER)                               0008077LINKHO 259
C E(N)=UNH2O(N)*12.38E-4/UNCO2(N)*P(N)                                  0008078LINKHO 260
      E(1;4608)=12.38E-4*UNH2O(1;4608)                                  0008079LINKHO 261
      E(1;4608)=E(1;4608)/UNCO2(1;4608)                                 0008080LINKHO 262
      E(1;4608)=E(1;4608)*P(1;4608)                                     0008081LINKHO 263
C**** COMPUTE TE (LOGARITHMIC INTERPOLATION)                             0008082LINKHO 264
      TH(1;3456)=TL(1;3456)/PLK(1;3456)                                 0008083LINKHO 265
C                                                                       0008084LINKHO 266
C    USE EXPBYK APPROXIMATION TO GET **.286                             0008085LINKHO 267
C                                                                       0008086LINKHO 268
      DO 66 L=2,NLAY                                                    0008087LINKHO 269
      KL=(L-1)*384+1                                                    0008088LINKHO 270
      KL1=KL-384                                                        0008089LINKHO 271
      CALL EXPBYK2(PLKE(KL1),PLE(KL),L)                                 0008090LINKHO 272
   66 CONTINUE                                                          0008091LINKHO 273
      XX1(1;3072)=PLKE(1;3072)-PLK(385;3072)                            0008092LINKHO 274
      XX1(1;3072)=XX1(1;3072)*TH(1;3072)                                0008093LINKHO 275
      XX2(1;3072)=PLK(1;3072)-PLKE(1;3072)                              0008094LINKHO 276
      XX2(1;3072)=XX2(1;3072)*TH(385;3072)                              0008095LINKHO 277
      XX1(1;3072)=XX1(1;3072)+XX2(1;3072)                               0008096LINKHO 278
      XX1(1;3072)=XX1(1;3072)*PLKE(1;3072)                              0008097LINKHO 279
      XX2(1;3072)=PLK(1;3072)-PLK(385;3072)                             0008098LINKHO 280
      TE(1537;3072)=XX1(1;3072)/XX2(1;3072)                             0008099LINKHO 281
C**** TE FOR STRATOSPHERE                                                0008100LINKHO 282
      TE(385;384)=TSTR(1;384)+TSTR(385;384)                             0008101LINKHO 283
      TE(385;384)=TE(385;384)/2.                                        0008102LINKHO 284
      TE(769;384)=TSTR(385;384)+TSTR(769;384)                           0008103LINKHO 285
      TE(769;384)=TE(769;384)/2.                                        0008104LINKHO 286
      TE(1;384)=2.*TSTR(1;384)                                          0008105LINKHO 287
      TE(1;384)=TE(1;384)-TE(385;384)                                   0008106LINKHO 288
      TE(1153;384)=TSTR(769;384)+TL(1;384)                              0008107LINKHO 289
      TE(1153;384)=TE(1153;384)/2.                                      0008108LINKHO 290
C**** TE NEAR GROUND                                                     0008109LINKHO 291
      TE(4609;384)=TS(1;384)                                            0008110LINKHO 292
      TE(4993;384)=TG(1;384)                                            0008111LINKHO 293
C**** CO2 PARTIAL PRESSURE FROM DP                                       0008112LINKHO 294
      UNCO2(1;4608)=CO2CM*UNCO2(1;4608)                                 0008113LINKHO 295
C**** CLOUD ARRAY                                                        0008114LINKHO 296
      NCLOUD(1;1152)=0                                                  0008115LINKHO 297
      NCLOUD(1153;3456)=CLOUD(1;3456)                                   0008116LINKHO 298
C****                                                                   0008117LINKHO 299
C**** BAND LOOP                                                          0008118LINKHO 300
C****                                                                   0008119LINKHO 301
      FE(1;4992)=0.                                                     0008120LINKHO 302
      FLXONG(1;384)=0.                                                  0008121LINKHO 303
      DO 200 LAM=1,12                                                   0008122LINKHO 304
C**** CALCULATE OPTICAL THICKNESS TAU(N,K)                               0008123LINKHO 305
      DO 2 N=1,NLAYRS                                                   0008124LINKHO 306
      NPTR=(N-1)*384+1                                                  0008125LINKHO 307
      IF (N.LE.3) TN(1;384)=TSTR(NPTR;384)/273.                         0008126LINKHO 308
      IF (N.GE.4) TN(1;384)=TL((N-4)*384+1;384)/273.                    0008127LINKHO 309
      TNSQ(1;384)=TN(1;384)*TN(1;384)                                   0008128LINKHO 310
C     A=FTEMP(A1(LAM,1),A1(LAM,2),A1(LAM,3),TN)                         0008129LINKHO 311
      A(1;384)=A1(LAM,3)*TNSQ(1;384)                                    0008130LINKHO 312
      X1(1;384)=A1(LAM,2)*TN(1;384)                                     0008131LINKHO 313
      A(1;384)=A(1;384)+X1(1;384)                                       0008132LINKHO 314
      A(1;384)=A(1;384)+A1(LAM,1)                                       0008133LINKHO 315
C     AAA=FTEMP(A2(LAM,1),A2(LAM,2),A2(LAM,3),TN)                       0008134LINKHO 316
```

```
      AAA(1;384)=A2(LAM,3)*TNSQ(1;384)                                  0008135LINKHO 317
      X1(1;384)=A2(LAM,2)*TN(1;384)                                     0008136LINKHO 318
      AAA(1;384)=AAA(1;384)+X1(1;384)                                   0008137LINKHO 319
      AAA(1;384)=AAA(1;384)+A2(LAM,1)                                   0008138LINKHO 320
      GO TO (3,3,3,3,20,20,3,3,20,3,3,3),LAM                            0008139LINKHO 321
C**** OVERLAPPING REGION                                                0008140LINKHO 322
C**** WATER VAPOR SELECTIVE ABSORPTION                                  0008141LINKHO 323
20    JJ=LAM-4                                                          0008142LINKHO 324
      IF(LAM.EQ.9) JJ=LAM-6                                             0008143LINKHO 325
C     SE=FTEMP(SH20(JJ,1),SH20(JJ,2),SH20(JJ,3),TN)                     0008144LINKHO 326
      SE(1;384)=SH20(JJ,3)*TNSQ(1;384)                                  0008145LINKHO 327
      X1(1;384)=SH20(JJ,2)*TN(1;384)                                    0008146LINKHO 328
      SE(1;384)=SE(1;384)+X1(1;384)                                     0008147LINKHO 329
      SE(1;384)=SE(1;384)+SH20(JJ,1)                                    0008148LINKHO 330
C     BE=FTEMP(BH20(JJ,1),BH20(JJ,2),BH20(JJ,3),TN)*PN                  0008149LINKHO 331
      BE(1;384)=BH20(JJ,3)*TNSQ(1;384)                                  0008150LINKHO 332
      X1(1;384)=BH20(JJ,2)*TN(1;384)                                    0008151LINKHO 333
      BE(1;384)=BE(1;384)+X1(1;384)                                     0008152LINKHO 334
      BE(1;384)=BE(1;384)+BH20(JJ,1)                                    0008153LINKHO 335
      BE(1;384)=BE(1;384)*P(NPTR;384)                                   0008154LINKHO 336
      X1(1;384)=SE(1;384)*UNH20(NPTR;384)                               0008155LINKHO 337
      X1(1;384)=X1(1;384)/BE(1;384)                                     0008156LINKHO 338
      L1(1;384)=X1(1;384).LT.1.E-2                                      0008157LINKHO 339
C IF(NOT L1) TAU=2.*BE*(SQRT(1+X1)-1)                                   0008158LINKHO 340
      TAU(1;384)=2.*BE(1;384)                                           0008159LINKHO 341
      X1(1;384)=X1(1;384)+1.                                            0008160LINKHO 342
      X1(1;384)=VSQRT(X1(1;384);X1(1;384))                              0008161LINKHO 343
      X1(1;384)=X1(1;384)-1.                                            0008162LINKHO 344
      TAU(1;384)=TAU(1;384)*X1(1;384)                                   0008163LINKHO 345
C IF(L1)TAU=SE*UNH20(N)                                                 0008164LINKHO 346
      X1(1;384)=SE(1;384)*UNH20(NPTR;384)                               0008165LINKHO 347
      TAU(1;384)=Q8VCTRL(X1(1;384),L1(1;384);TAU(1;384))                0008166LINKHO 348
5     GO TO (6,6,21),JJ                                                 0008167LINKHO 349
C**** OZONE SELECTIVE ABSORPTION                                        0008168LINKHO 350
C21   SE1=12.7892-14.3689*TN+7.3921*TN*TN                               0008169LINKHO 351
21    SE(1;384)=7.3921*TNSQ(1;384)                                      0008170LINKHO 352
      X1(1;384)=14.3689*TN(1;384)                                       0008171LINKHO 353
      SE(1;384)=SE(1;384)-X1(1;384)                                     0008172LINKHO 354
      SE(1;384)=SE(1;384)+12.7892                                       0008173LINKHO 355
C     BE1=(1.0635+1.9570*TN-.3227*TN*TN)*PN                             0008174LINKHO 356
      BE(1;384)=-.3227*TNSQ(1;384)                                      0008175LINKHO 357
      X1(1;384)=1.9570*TN(1;384)                                        0008176LINKHO 358
      BE(1;384)=BE(1;384)+X1(1;384)                                     0008177LINKHO 359
      BE(1;384)=BE(1;384)+1.0635                                        0008178LINKHO 360
      BE(1;384)=BE(1;384)*P(NPTR;384)                                   0008179LINKHO 361
      UN1(1;384)=UNO3(NPTR;384)                                         0008180LINKHO 362
      GO TO 7                                                           0008181LINKHO 363
C**** CARBON DIOXIDE SELECTIVE ABSORPTION                                0008182LINKHO 364
C6    SE1=7.4197-6.9225*TN+4.2328*TN*TN                                 0008183LINKHO 365
6     SE(1;384)=4.2328*TNSQ(1;384)                                      0008184LINKHO 366
      X1(1;384)=6.9225*TN(1;384)                                        0008185LINKHO 367
      SE(1;384)=SE(1;384)-X1(1;384)                                     0008186LINKHO 368
      SE(1;384)=SE(1;384)+7.4197                                        0008187LINKHO 369
C     BE1=(0.1697-0.1734*TN+0.2410*TN*TN)*PN                            0008188LINKHO 370
      BE(1;384)=.2410*TNSQ(1;384)                                       0008189LINKHO 371
      X1(1;384)=.1734*TN(1;384)                                         0008190LINKHO 372
      BE(1;384)=BE(1;384)-X1(1;384)                                     0008191LINKHO 373
      BE(1;384)=BE(1;384)+.1697                                         0008192LINKHO 374
      BE(1;384)=BE(1;384)*P(NPTR;384)                                   0008193LINKHO 375
      IF(LAM.NE.6) GO TO 666                                            0008194LINKHO 376
      SE(1;384)=.0815*TNSQ(1;384)                                       0008195LINKHO 377
      X1(1;384)=.1113*TN(1;384)                                         0008196LINKHO 378
      SE(1;384)=SE(1;384)-X1(1;384)                                     0008197LINKHO 379
```

```
      SE(1;384)=SE(1;384)+.0392                                      0008198LINKHO 380
      BE(1;384)=.0297*TNSQ(1;384)                                    0008199LINKHO 381
      X1(1;384)=.2077*TN(1;384)                                      0008200LINKHO 382
      BE(1;384)=BE(1;384)+X1(1;384)                                  0008201LINKHO 383
      BE(1;384)=BE(1;384)-.0651                                      0008202LINKHO 384
      BE(1;384)=BE(1;384)*P(NPTR;384)                                0008203LINKHO 385
666   CONTINUE                                                       0008204LINKHO 386
      UN1(1;384)=UNCO2(NPTR;384)                                     0008205LINKHO 387
C7    XX=SE1*UN1/BE1                                                 0008206LINKHO 388
7     X1(1;384)=SE(1;384)*UN1(1;384)                                 0008207LINKHO 389
      X1(1;384)=X1(1;384)/BE(1;384)                                  0008208LINKHO 390
      L1(1;384)=X1(1;384).LT.1.E-2                                   0008209LINKHO 391
C IF(.NOT.L1)                                                        0008210LINKHO 392
      TAU1(1;384)=2.*BE(1;384)                                       0008211LINKHO 393
      X1(1;384)=X1(1;384)+1.                                         0008212LINKHO 394
      X1(1;384)=VSQRT(X1(1;384);X1(1;384))                           0008213LINKHO 395
      X1(1;384)=X1(1;384)-1.                                         0008214LINKHO 396
      TAU1(1;384)=TAU1(1;384)*X1(1;384)                              0008215LINKHO 397
C IF(L1)                                                             0008216LINKHO 398
      X1(1;384)=SE(1;384)*UN1(1;384)                                 0008217LINKHO 399
      TAU1(1;384)=Q8VCTRL(X1(1;384),L1(1;384);TAU1(1;384))           0008218LINKHO 400
      TAUT(1;384)=TAU(1;384)+TAU1(1;384)                             0008219LINKHO 401
      DO 14 K=1,3                                                    0008220LINKHO 402
C                                                                    0008221LINKHO 403
      KPTR=(K-1)*384+1                                               0008222LINKHO 404
      NKPTR=(K-1)*4608+NPTR                                          0008223LINKHO 405
      AA(1;384)=A(1;384)*A3(LAM,K)                                   0008224LINKHO 406
      BB(1;384)=AAA(1;384)*A4(LAM,K)                                 0008225LINKHO 407
C     FKGAS(K)=AA*PN/(1.+BB*PN)                                      0008226LINKHO 408
      X1(1;384)=BB(1;384)*P(NPTR;384)                                0008227LINKHO 409
      X1(1;384)=X1(1;384)+1.                                         0008228LINKHO 410
      FKGAS(KPTR;384)=AA(1;384)*P(NPTR;384)                          0008229LINKHO 411
      FKGAS(KPTR;384)=FKGAS(KPTR;384)/X1(1;384)                      0008230LINKHO 412
      IF(LAM.EQ.9) GO TO 15                                          0008231LINKHO 413
C     AA=FTEMP(B1(K,1,JJ),B1(K,2,JJ),B1(K,3,JJ),TN)                  0008232LINKHO 414
      AA(1;384)=B1(K,3,JJ)*TNSQ(1;384)                               0008233LINKHO 415
      X1(1;384)=B1(K,2,JJ)*TN(1;384)                                 0008234LINKHO 416
      AA(1;384)=AA(1;384)+X1(1;384)                                  0008235LINKHO 417
      AA(1;384)=AA(1;384)+B1(K,1,JJ)                                 0008236LINKHO 418
      FKGAS2(KPTR;384)=AA(1;384)*P(NPTR;384)                         0008237LINKHO 419
C     BB=FTEMP(B2(K,1,JJ),B2(K,2,JJ),B2(K,3,JJ),TN)                  0008238LINKHO 420
      BB(1;384)=B2(K,3,JJ)*TNSQ(1;384)                               0008239LINKHO 421
      X1(1;384)=B2(K,2,JJ)*TN(1;384)                                 0008240LINKHO 422
      BB(1;384)=BB(1;384)+X1(1;384)                                  0008241LINKHO 423
      BB(1;384)=BB(1;384)+B2(K,1,JJ)                                 0008242LINKHO 424
C     CC=FTEMP(B3(K,1,JJ),B3(K,2,JJ),B3(K,3,JJ),TN)                  0008243LINKHO 425
      CC(1;384)=B3(K,3,JJ)*TNSQ(1;384)                               0008244LINKHO 426
      X1(1;384)=B3(K,2,JJ)*TN(1;384)                                 0008245LINKHO 427
      CC(1;384)=CC(1;384)+X1(1;384)                                  0008246LINKHO 428
      CC(1;384)=CC(1;384)+B3(K,1,JJ)                                 0008247LINKHO 429
C     FKGAS2(K)=AA*PN/(1.+BB*PN**CC)                                 0008248LINKHO 430
      X1(1;384)=P(NPTR;384)**CC(1;384)                               0008249LINKHO 431
      X1(1;384)=X1(1;384)*BB(1;384)                                  0008250LINKHO 432
      X1(1;384)=X1(1;384)+1.                                         0008251LINKHO 433
      FKGAS2(KPTR;384)=AA(1;384)*P(NPTR;384)                         0008252LINKHO 434
      FKGAS2(KPTR;384)=FKGAS2(KPTR;384)/X1(1;384)                    0008253LINKHO 435
      GO TO 161                                                      0008254LINKHO 436
15    GO TO (23,23,17),K                                             0008255LINKHO 437
C23   AA=FTEMP(C1(K,1),C1(K,2),C1(K,3),TN)                           0008256LINKHO 438
23    AA(1;384)=C1(K,3)*TNSQ(1;384)                                  0008257LINKHO 439
      X1(1;384)=C1(K,2)*TN(1;384)                                    0008258LINKHO 440
      AA(1;384)=AA(1;384)+X1(1;384)                                  0008259LINKHO 441
      AA(1;384)=AA(1;384)+C1(K,1)                                    0008260LINKHO 442
```

```
C     BB=FTEMP(C2(K,1),C2(K,2),C2(K,3),TN)
      BB(1;384)=C2(K,3)*TNSQ(1;384)
      X1(1;384)=C2(K,2)*TN(1;384)
      BB(1;384)=BB(1;384)+X1(1;384)
      BB(1;384)=BB(1;384)+C2(K,1)
C     FKGAS2(K)=AA*PN/(1.+BB*PN)
      FKGAS2(KPTR;384)=AA(1;384)*P(NPTR;384)
      X1(1;384)=BB(1;384)*P(NPTR;384)
      X1(1;384)=X1(1;384)+1.
      FKGAS2(KPTR;384)=FKGAS2(KPTR;384)/X1(1;384)
      GO TO 161
C17   AA=24.0109-27.9638*TN+13.757*TN*TN
17    AA(1;384)=13.757*TNSQ(1;384)
      X1(1;384)=27.9638*TN(1;384)
      AA(1;384)=AA(1;384)-X1(1;384)
      AA(1;384)=AA(1;384)+24.0109
C     BB=0.1716+0.06669*TN-0.02572*TN*TN
C     FOR CONVENIENCE, COMPUTE -BB
      BB(1;384)=.02572*TNSQ(1;384)
      X1(1;384)=.06669*TN(1;384)
      BB(1;384)=BB(1;384)-X1(1;384)
      BB(1;384)=BB(1;384)-.1716
C     FKGAS2(K)=AA*PN**(-BB)
      X2(1;384)=P(NPTR;384)**BB(1;384)
      FKGAS2(KPTR;384)=AA(1;384)*X2(1;384)
C16   TAUN(N,K)=(FKGAS(K)*UN*TAU+FKGAS2(K)*UN1*TAU1)/(TAUT+1.E-40)
161   FKGAS(KPTR;384)=FKGAS(KPTR;384)*UNH2O(NPTR;384)
      FKGAS(KPTR;384)=FKGAS(KPTR;384)*TAU(1;384)
      FKGAS2(KPTR;384)=FKGAS2(KPTR;384)*UN1(1;384)
      FKGAS2(KPTR;384)=FKGAS2(KPTR;384)*TAU1(1;384)
      X1(1;384)=TAUT(1;384)+1.E-40
      TAUN(NKPTR;384)=FKGAS(KPTR;384)+FKGAS2(KPTR;384)
16    TAUN(NKPTR;384)=TAUN(NKPTR;384)/X1(1;384)
14    CONTINUE
      GO TO 13
3     DO 10 K=1,3
      KPTR=(K-1)*384+1
      AA(1;384)=A(1;384)*A3(LAM,K)
      BB(1;384)=AAA(1;384)*A4(LAM,K)
C10   FKGAS(K)=AA*PN/(1.0+BB*PN)
      FKGAS(KPTR;384)=AA(1;384)*P(NPTR;384)
      X1(1;384)=BB(1;384)*P(NPTR;384)
      X1(1;384)=X1(1;384)+1.
      FKGAS(KPTR;384)=FKGAS(KPTR;384)/X1(1;384)
10    CONTINUE
      DO 12 K=1,3
      KPTR=(K-1)*384+1
      NKPTR=(K-1)*4608+NPTR
12    TAUN(NKPTR;384)=FKGAS(KPTR;384)*UNH2O(NPTR;384)
13    CONTINUE
      GO TO (1,1,27,27,1,22,22,22,22,22,1,1),LAM
C**** WATER VAPOR CONTINUUM ABSORPTION
22    JJ=LAM-5
C XK2=FTEMP(WK(JJ,1),WK(JJ,2),WK(JJ,3),TN)
      X2(1;384)=WK(JJ,3)*TNSQ(1;384)
      X1(1;384)=WK(JJ,2)*TN(1;384)
      X2(1;384)=X2(1;384)+X1(1;384)
      X2(1;384)=X2(1;384)+WK(JJ,1)
C WK1=.005*XK2
      X1(1;384)=.005*X2(1;384)
      GO TO 29
27    JJ=LAM-2
      X1(1;384)=WK20(JJ,1)
```

```
      X2(1;384)=WK20(JJ,2)                                                     0008324LINKHO 506
29    CONTINUE                                                                 0008325LINKHO 507
C     XK=XK1*PN+XK2*E(N)                                                       0008326LINKHO 508
      X1(1;384)=X1(1;384)*P(NPTR;384)                                          0008327LINKHO 509
      X2(1;384)=X2(1;384)*E(NPTR;384)                                          0008328LINKHO 510
      X2(1;384)=X2(1;384)+X1(1;384)                                            0008329LINKHO 511
      X2(1;384)=X2(1;384)*UNH20(NPTR;384)                                      0008330LINKHO 512
      DO 18 K=1,3                                                              0008331LINKHO 513
      NKPTR=(K-1)*4608+NPTR                                                    0008332LINKHO 514
18    TAUN(NKPTR;384)=TAUN(NKPTR;384)+X2(1;384)                                0008333LINKHO 515
1     CONTINUE                                                                 0008334LINKHO 516
28    CONTINUE                                                                 0008335LINKHO 517
      DO 11 K=1,3                                                              0008336LINKHO 518
      KPTR=(K-1)*384+1                                                         0008337LINKHO 519
      NKPTR=(K-1)*4608+NPTR                                                    0008338LINKHO 520
      TAUN(NKPTR;384)=TAUN(NKPTR;384)+TA(LAM,N)                                0008339LINKHO 521
11    CONTINUE                                                                 0008340LINKHO 522
2     CONTINUE                                                                 0008341LINKHO 523
C**** COMPUTE PLANCK FUNCTION AT EACH LAYER                                    0008342LINKHO 524
      DO 95 N=1,NG1                                                            0008343LINKHO 525
      NPTR=(N-1)*384+1                                                         0008344LINKHO 526
      X1(1;384)=PF2(LAM)                                                       0008345LINKHO 527
      X1(1;384)=X1(1;384)/TE(NPTR;384)                                         0008346LINKHO 528
      X1(1;384)=VEXP(X1(1;384);X1(1;384))                                      0008347LINKHO 529
      X1(1;384)=X1(1;384)-1.                                                   0008348LINKHO 530
      BTOP(NPTR;384)=PF1(LAM)                                                  0008349LINKHO 531
95    BTOP(NPTR;384)=BTOP(NPTR;384)/X1(1;384)                                  0008350LINKHO 532
      DO 100 K=1,3                                                             0008351LINKHO 533
C**** ADDING LOOP                                                              0008352LINKHO 534
C**** COEFFICIENT IN SUMMATION OF HEATING OVER MAGNITUDES AND BANDS             0008353LINKHO 535
      CKLAM=COEK(K)*COELAM(LAM)                                                0008354LINKHO 536
C**** INITIALIZE CUMULATIVE OPTICAL DEPTH                                      0008355LINKHO 537
      TAU(1;384)=0.                                                            0008356LINKHO 538
C**** FLAG FOR CLOUDS                                                          0008357LINKHO 539
      CLDFLG(1;384)=CLDFLG(1;384).XOR.CLDFLG(1;384)                            0008358LINKHO 540
C**** FLAG FOR AEROSOLS (OR CLOUDS TREATED AS AEROSOLS)                        0008359LINKHO 541
      AERFLG(1;384)=AERFLG(1;384).XOR.AERFLG(1;384)                            0008360LINKHO 542
C**** CUMULATIVE UPWARDS EMISSION FOR TOP COMPOSITE REGION                     0008361LINKHO 543
C     EUPCN=0.                                                                 0008362LINKHO 544
C**** CUMULATIVE DOWNWARDS EMISSION FOR TOP COMPOSITE REGION                   0008363LINKHO 545
      EDNCN(1;384)=0.                                                          0008364LINKHO 546
C**** COMPOSITE TRANSMISSION                                                   0008365LINKHO 547
      TDFCN(1;384)=1.                                                          0008366LINKHO 548
C**** COMPOSITE DOWNWARDS REFLECTION                                           0008367LINKHO 549
      RDNCN(1;384)=0.                                                          0008368LINKHO 550
C****                                                                          0008369LINKHO 551
C**** CUMULATIVE QUANTITIES ARE COMPUTED IN EDNCN/TDFCN/RDNCN BUT               0008370LINKHO 552
C**** PERIODICALLY STORED INTO EUPC,EDNC,TDFC,ETC. BY LAYER POSITION           0008371LINKHO 553
C****                                                                          0008372LINKHO 554
      DO 101 N=1,NLAYRS                                                        0008373LINKHO 555
      NPTR=(N-1)*384+1                                                         0008374LINKHO 556
      NKPTR=(K-1)*4608+NPTR                                                    0008375LINKHO 557
C****                                                                          0008376LINKHO 558
C**** SINGLE LAYER COMPUTATION                                                 0008377LINKHO 559
C**** EUP=UPWARDS EMISSION                                                     0008378LINKHO 560
C**** EDN=DOWNWARDS EMISSION                                                   0008379LINKHO 561
C**** TDF=TRANSMISSION                                                         0008380LINKHO 562
C**** REF=REFLECTION                                                           0008381LINKHO 563
C****                                                                          0008382LINKHO 564
      TAUCIR(1;384)=CIREXT(LAM)                                                0008383LINKHO 565
      TAUCIR(1;384)=TAUCIR(1;384)*CTAUSS                                       0008384LINKHO 566
      TAUCIR(1;384)=TAUCIR(1;384)*NCLOUD(NPTR;384)                             0008385LINKHO 567
      TAUN(NKPTR;384)=TAUN(NKPTR;384)+TAUCIR(1;384)                            0008386LINKHO 568
```

```
C***********************************************************************0008387LINKHO 569
C SET PI0 TO ZERO FOR DARK CLOUDS                                        0008388LINKHO 570
C***********************************************************************0008389LINKHO 571
      X1(1;384)=TAUN(NKPTR;384)+1.E-40                                  0008390LINKHO 572
      X2(1;384)=TAUCIR(1;384)*PICIRO(LAM)                               0008391LINKHO 573
      X2(1;384)=X2(1;384)+PIZ(LAM,N)                                    0008392LINKHO 574
      PI0(1;384)=X2(1;384)/X1(1;384)                                    0008393LINKHO 575
      IF(N.LE.3) TN(1;384)=TSTR(NPTR;384)/273.                          0008394LINKHO 576
      IF(N.GE.4) TN(1;384)=TL((N-4)*384+1;384)/273.                     0008395LINKHO 577
C     IF(TN.GE..85348.AND.NCLOUD(N).GT.0)PI0=0.                         0008396LINKHO 578
      L1(1;384)=TN(1;384).GE..85348                                     0008397LINKHO 579
      L2(1;384)=NCLOUD(NPTR;384).GT.0                                   0008398LINKHO 580
      L1(1;384)=L1(1;384).AND.L2(1;384)                                 0008399LINKHO 581
      PI0(1;384)=Q8VCTRL(ZERO(1;384),L1(1;384);PI0(1;384))              0008400LINKHO 582
C**** EMISSION CALCULATIONS FOR HAZE LAYER.                              0008401LINKHO 583
C**** EXACT IN THE SENSE OF ISOTROPIC SCATTERING                         0008402LINKHO 584
C**** EXACT SOLUTION=TWO-STREAM SOLUTION*FORGE FACTOR(PI0,TAU0)           0008403LINKHO 585
      L1(1;384)=PI0(1;384).GT.1.E-4                                     0008404LINKHO 586
      LOUT=Q8SCNT(L1(1;384))                                            0008405LINKHO 587
      IF (LOUT.EQ.0) GO TO 165                                          0008406LINKHO 588
      PI0C(1;LOUT)=Q8VCMPRS(PI0(1;384),L1(1;384);PI0C(1;LOUT))          0008407LINKHO 589
      TAUNC(1;LOUT)=Q8VCMPRS(TAUN(NKPTR;384),L1(1;384);TAUNC(1;LOUT))   0008408LINKHO 590
      BTOPN(1;LOUT)=Q8VCMPRS(BTOP(NPTR;384),L1(1;384);BTOPN(1;LOUT))    0008409LINKHO 591
      BTOPNP(1;LOUT)=Q8VCMPRS(BTOP(NPTR+384;384),L1(1;384);             0008410LINKHO 592
     * BTOPNP(1;LOUT))                                                  0008411LINKHO 593
      AER1(1;LOUT)=ONE(1;LOUT)-PI0C(1;LOUT)                             0008412LINKHO 594
      X1(1;LOUT)=PI0C(1;LOUT)*CB(LAM,N)                                 0008413LINKHO 595
      AER2(1;LOUT)=ONE(1;LOUT)-X1(1;LOUT)                               0008414LINKHO 596
      X1(1;LOUT)=AER1(1;LOUT)/AER2(1;LOUT)                              0008415LINKHO 597
      AERA(1;LOUT)=VSQRT(X1(1;LOUT);AERA(1;LOUT))                       0008416LINKHO 598
      AERU(1;LOUT)=ONE(1;LOUT)-AERA(1;LOUT)                             0008417LINKHO 599
      AERU(1;LOUT)=AERU(1;LOUT)/2.                                      0008418LINKHO 600
      AERV(1;LOUT)=ONE(1;LOUT)+AERA(1;LOUT)                             0008419LINKHO 601
      AERV(1;LOUT)=AERV(1;LOUT)/2.0                                     0008420LINKHO 602
      AERC(1;LOUT)=3.0*AER1(1;LOUT)                                     0008421LINKHO 603
      AERC(1;LOUT)=AERC(1;LOUT)*AER2(1;LOUT)                            0008422LINKHO 604
      AERC(1;LOUT)=VSQRT(AERC(1;LOUT);AERC(1;LOUT))                     0008423LINKHO 605
      X1(1;LOUT)=AERC(1;LOUT)*TAUNC(1;LOUT)                             0008424LINKHO 606
      X1(1;LOUT)=-X1(1;LOUT)                                            0008425LINKHO 607
      EX1(1;LOUT)=VEXP(X1(1;LOUT);EX1(1;LOUT))                          0008426LINKHO 608
C..... TEMPORARY TRAP FOR UNDERFLOWS (AS IN SCALAR CODE)                  0008427LINKHO 609
      TSTEXP(1;LOUT) = X1(1;LOUT) .LT. -180.218                         0008428LINKHO 610
      EX1(1;LOUT) = Q8VCTRL(ZERO(1;LOUT),TSTEXP(1;LOUT);EX1(1;LOUT))    0008429LINKHO 611
      TSTEXP(1;LOUT) = EX1(1;LOUT) .LT. 1.E-30                          0008430LINKHO 612
      EX1(1;LOUT) = Q8VCTRL(ZERO(1;LOUT),TSTEXP(1;LOUT);EX1(1;LOUT))    0008431LINKHO 613
      EX2(1;LOUT)=EX1(1;LOUT)*EX1(1;LOUT)                               0008432LINKHO 614
C**** FORGE FACTOR FOR ISOTROPIC SCATTERING                              0008433LINKHO 615
      X1(1;LOUT)=AERV(1;LOUT)*AERV(1;LOUT)                              0008434LINKHO 616
      X2(1;LOUT)=AERU(1;LOUT)*AERU(1;LOUT)                              0008435LINKHO 617
      X2(1;LOUT)=X2(1;LOUT)*EX2(1;LOUT)                                 0008436LINKHO 618
      DENO(1;LOUT)=X1(1;LOUT)-X2(1;LOUT)                                0008437LINKHO 619
      DNM0(1;LOUT)=BTOPN(1;LOUT)-BTOPNP(1;LOUT)                         0008438LINKHO 620
      DNM0(1;LOUT)=DNM0(1;LOUT)/TAUNC(1;LOUT)                           0008439LINKHO 621
      DNM0(1;LOUT)=DNM0(1;LOUT)/AERC(1;LOUT)                            0008440LINKHO 622
      X2(1;LOUT)=AERU(1;LOUT)*EX2(1;LOUT)                               0008441LINKHO 623
      X1(1;LOUT)=AERV(1;LOUT)-X2(1;LOUT)                                0008442LINKHO 624
      X2(1;LOUT)=AERA(1;LOUT)*EX1(1;LOUT)                               0008443LINKHO 625
      X1(1;LOUT)=X1(1;LOUT)-X2(1;LOUT)                                  0008444LINKHO 626
      DNM0(1;LOUT)=DNM0(1;LOUT)*X1(1;LOUT)                              0008445LINKHO 627
      X1(1;LOUT)=AERU(1;LOUT)*EX2(1;LOUT)                               0008446LINKHO 628
      DNM1(1;LOUT)=AERV(1;LOUT)+X1(1;LOUT)                              0008447LINKHO 629
C      EUP(N)=(BTOP(N)*DNM1-DNM0-BTOP(N+1)*EX1)/DENO*FTWO*AERA          0008448LINKHO 630
C**** USE TAUNC FOR TEMPORARY STORAGE                                    0008449LINKHO 631
```

```
      X1(1;LOUT)=BTOPN(1;LOUT)*DNM1(1;LOUT)                              0008450LINKHO 632
      X1(1;LOUT)=X1(1;LOUT)-DNM0(1;LOUT)                                 0008451LINKHO 633
      X2(1;LOUT)=BTOPNP(1;LOUT)*EX1(1;LOUT)                              0008452LINKHO 634
      TAUNC(1;LOUT)=X1(1;LOUT)-X2(1;LOUT)                                0008453LINKHO 635
      TAUNC(1;LOUT)=TAUNC(1;LOUT)/DENO(1;LOUT)                           0008454LINKHO 636
      TAUNC(1;LOUT)=TAUNC(1;LOUT)*AERA(1;LOUT)                           0008455LINKHO 637
      EUP(NPTR;384)=Q8VXPND(TAUNC(1;LOUT),L1(1;384);EUP(NPTR;384))       0008456LINKHO 638
C     EDN(N)=(BTOP(N+1)*DNM1+DNM0-BTOP(N)*EX1)/DENO*FTWO*AERA            0008457LINKHO 639
      TAUNC(1;LOUT)=BTOPNP(1;LOUT)*DNM1(1;LOUT)                          0008458LINKHO 640
      TAUNC(1;LOUT)=TAUNC(1;LOUT)+DNM0(1;LOUT)                           0008459LINKHO 641
      X1(1;LOUT)=BTOPN(1;LOUT)*EX1(1;LOUT)                               0008460LINKHO 642
      TAUNC(1;LOUT)=TAUNC(1;LOUT)-X1(1;LOUT)                             0008461LINKHO 643
      TAUNC(1;LOUT)=TAUNC(1;LOUT)/DENO(1;LOUT)                           0008462LINKHO 644
      TAUNC(1;LOUT)=TAUNC(1;LOUT)*AERA(1;LOUT)                           0008463LINKHO 645
      EDN(NPTR;384)=Q8VXPND(TAUNC(1;384),L1(1;384);EDN(NPTR;384))        0008464LINKHO 646
C**** REF(N),TDF(N) BASED ON TWO STREAM SOLUTION                          0008465LINKHO 647
C     REF(N)=AERU*AERV*(1.-EX2)/DENO                                     0008466LINKHO 648
      TAUNC(1;LOUT)=AERU(1;LOUT)*AERV(1;LOUT)                            0008467LINKHO 649
      X1(1;LOUT)=ONE(1;LOUT)-EX2(1;LOUT)                                 0008468LINKHO 650
      TAUNC(1;LOUT)=TAUNC(1;LOUT)*X1(1;LOUT)                             0008469LINKHO 651
      TAUNC(1;LOUT)=TAUNC(1;LOUT)/DENO(1;LOUT)                           0008470LINKHO 652
      REF(NPTR;384)=Q8VXPND(TAUNC(1;LOUT),L1(1;384);REF(NPTR;384))       0008471LINKHO 653
C     TDF(N)=(AERV-AERU)/DENO*EX1                                        0008472LINKHO 654
      TAUNC(1;LOUT)=AERV(1;LOUT)-AERU(1;LOUT)                            0008473LINKHO 655
      TAUNC(1;LOUT)=TAUNC(1;LOUT)/DENO(1;LOUT)                           0008474LINKHO 656
      TAUNC(1;LOUT)=TAUNC(1;LOUT)*EX1(1;LOUT)                            0008475LINKHO 657
      TDF(NPTR;384)=Q8VXPND(TAUNC(1;LOUT),L1(1;384);TDF(NPTR;384))       0008476LINKHO 658
165   CONTINUE                                                           0008477LINKHO 659
C****                                                                    0008478LINKHO 660
C**** DARK CLOUDS                                                         0008479LINKHO 661
C****                                                                    0008480LINKHO 662
C NEXT TEST ON CLOUDS                                                     0008481LINKHO 663
C BUT FIRST TEST EXCLUDES ALL OTHERS                                      0008482LINKHO 664
      L1(1;384)=.NOT.L1(1;384)                                           0008483LINKHO 665
      L3(1;384)=NCLOUD(NPTR;384).GT.0                                    0008484LINKHO 666
C L2=SECOND TEST=.NOT.FIRST.AND.SECOND                                    0008485LINKHO 667
      L2(1;384)=L1(1;384).AND.L3(1;384)                                  0008486LINKHO 668
C L1=.NOT.FIRST.AND..NOT.SECOND                                           0008487LINKHO 669
      L3(1;384)=.NOT.L2(1;384)                                           0008488LINKHO 670
      L1(1;384)=L1(1;384).AND.L3(1;384)                                  0008489LINKHO 671
      TDF(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);TDF(NPTR;384))         0008490LINKHO 672
      REF(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);REF(NPTR;384))         0008491LINKHO 673
      EDN(NPTR;384)=Q8VCTRL(BTOP(NPTR+384;384),L2(1;384);EDN(NPTR;384))  0008492LINKHO 674
      EUP(NPTR;384)=Q8VCTRL(BTOP(NPTR;384),L2(1;384);EUP(NPTR;384))      0008493LINKHO 675
C****                                                                    0008494LINKHO 676
C**** THICK LAYER                                                         0008495LINKHO 677
C****                                                                    0008496LINKHO 678
      L2(1;384)=TAUN(NKPTR;384).GT.15.                                   0008497LINKHO 679
      L2(1;384)=L2(1;384).AND.L1(1;384)                                  0008498LINKHO 680
      TDF(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);TDF(NPTR;384))         0008499LINKHO 681
      EXTAU(1;384)=Q8VCTRL(ZERO(1;384),L2(1;384);EXTAU(1;384))           0008500LINKHO 682
C****                                                                    0008501LINKHO 683
C**** TRANSPARENT LAYER                                                   0008502LINKHO 684
C****                                                                    0008503LINKHO 685
      L2(1;384)=TAUN(NKPTR;384).LT.1.E-4                                 0008504LINKHO 686
      L2(1;384)=L2(1;384).AND.L1(1;384)                                  0008505LINKHO 687
      TDF(NPTR;384)=Q8VCTRL(ONE(1;384),L2(1;384);TDF(NPTR;384))          0008506LINKHO 688
      REF(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);REF(NPTR;384))         0008507LINKHO 689
      EXTAU(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);EXTAU(NPTR;384))     0008508LINKHO 690
      EUP(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);EUP(NPTR;384))         0008509LINKHO 691
      EDN(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);EDN(NPTR;384))         0008510LINKHO 692
C****                                                                    0008511LINKHO 693
C**** INTERMEDIATE RANGE                                                  0008512LINKHO 694
```

```
C****                                                                   0008513LINKHO 695
      L2(1;384)=TAUN(NKPTR;384).LE.15.                                  0008514LINKHO 696
      L3(1;384)=TAUN(NKPTR;384).GE.1.E-4                                0008515LINKHO 697
      L2(1;384)=L2(1;384).AND.L3(1;384)                                 0008516LINKHO 698
      L2(1;384)=L2(1;384).AND.L1(1;384)                                 0008517LINKHO 699
      X1(1;384)=-TAUN(NKPTR;384)                                        0008518LINKHO 700
      EXTAU(1;384)=VEXP(X1(1;384);EXTAU(1;384))                         0008519LINKHO 701
      TY(1;384)=20.*TAUN(NKPTR;384)                                     0008520LINKHO 702
C**** PREVENT TABLE OVERFLOW BY STORING 1'S IN LOOK-UP VECTOR            0008521LINKHO 703
      ITY(1;384)=1                                                      0008522LINKHO 704
      I1(1;384)=TY(1;384)+1.0                                           0008523LINKHO 705
      RITY(1;384)=Q8VCTRL(RI1(1;384),L2(1;384);RITY(1;384))             0008524LINKHO 706
      X1(1;384)=Q8VGATHR(TE3(1;301),ITY(1;384);X1(1;384))               0008525LINKHO 707
      ITY(1;384)=ITY(1;384)+1                                           0008526LINKHO 708
      X2(1;384)=Q8VGATHR(TE3(1;301),ITY(1;384);X2(1;384))               0008527LINKHO 709
C     TDF(N)=X2-X1                                                      0008528LINKHO 710
      X3(1;384)=X2(1;384)-X1(1;384)                                     0008529LINKHO 711
      X2(1;384)=TY(1;384)-ITY(1;384)                                    0008530LINKHO 712
      X2(1;384)=X2(1;384)+2.                                            0008531LINKHO 713
C     TDF(N)=TDF(N)*X2                                                  0008532LINKHO 714
      X3(1;384)=X3(1;384)*X2(1;384)                                     0008533LINKHO 715
C     TDF(N)=TDF(N)+X1                                                  0008534LINKHO 716
      X3(1;384)=X3(1;384)+X1(1;384)                                     0008535LINKHO 717
C     CONTROLLED STORE INTO TDF(N)                                      0008536LINKHO 718
      TDF(NPTR;384)=Q8VCTRL(X3(1;384),L2(1;384);TDF(NPTR;384))          0008537LINKHO 719
C****                                                                   0008538LINKHO 720
C**** CALCULATIONS COMMON TO INTERMEDIATE AND HIGH RANGE                 0008539LINKHO 721
C****                                                                   0008540LINKHO 722
      L2(1;384)=L3(1;384).AND.L1(1;384)                                 0008541LINKHO 723
      REF(NPTR;384)=Q8VCTRL(ZERO(1;384),L2(1;384);REF(NPTR;384))        0008542LINKHO 724
C     DFB=(BTOP(N)-BTOP(N+1))*6.6667E-01                                0008543LINKHO 725
      X1(1;384)=BTOP(NPTR;384)-BTOP(NPTR+384;384)                       0008544LINKHO 726
      X1(1;384)=X1(1;384)*6.6667E-01                                    0008545LINKHO 727
C     FGRAD=DFB*((1.0-EXTAU)/X-TDF(N))                                  0008546LINKHO 728
      X2(1;384)=ONE(1;384)-EXTAU(1;384)                                 0008547LINKHO 729
      X2(1;384)=X2(1;384)/TAUN(NKPTR;384)                               0008548LINKHO 730
      X2(1;384)=X2(1;384)-TDF(NPTR;384)                                 0008549LINKHO 731
      X2(1;384)=X2(1;384)*X1(1;384)                                     0008550LINKHO 732
C     ANS=1.0-TDF(N)                                                    0008551LINKHO 733
      X1(1;384)=ONE(1;384)-TDF(NPTR;384)                                0008552LINKHO 734
C     EDN(N)=BTOP(N+1)*ANS+FGRAD                                        0008553LINKHO 735
      X3(1;384)=BTOP(NPTR+384;384)*X1(1;384)                            0008554LINKHO 736
      X3(1;384)=X3(1;384)+X2(1;384)                                     0008555LINKHO 737
      EDN(NPTR;384)=Q8VCTRL(X3(1;384),L2(1;384);EDN(NPTR;384))          0008556LINKHO 738
C     EUP(N)=BTOP(N)*ANS-FGRAD                                          0008557LINKHO 739
      X3(1;384)=BTOP(NPTR;384)*X1(1;384)                                0008558LINKHO 740
      X3(1;384)=X3(1;384)-X2(1;384)                                     0008559LINKHO 741
      EUP(NPTR;384)=Q8VCTRL(X3(1;384),L2(1;384);EUP(NPTR;384))          0008560LINKHO 742
C****                                                                   0008561LINKHO 743
C**** FORM TOP COMPOSITE LAYER (ADDITION)                                0008562LINKHO 744
C****                                                                   0008563LINKHO 745
C109  DENO=1.0-RDNCN*REF(N)                                             0008564LINKHO 746
  109 X1(1;384)=RDNCN(1;384)*REF(NPTR;384)                              0008565LINKHO 747
      DENO(1;384)=ONE(1;384)-X1(1;384)                                  0008566LINKHO 748
C     EUPCN=EUPCN+(EUP(N)+EDNCN*REF(N))*TOFC(N)/DENO                    0008567LINKHO 749
C     EDNCN=EDN(N)+(EDNCN+EUP(N)*RDNCN)*TDF(N)/DENO                     0008568LINKHO 750
      X1(1;384)=EUP(NPTR;384)*RDNCN(1;384)                              0008569LINKHO 751
      EDNCN(1;384)=EDNCN(1;384)+X1(1;384)                               0008570LINKHO 752
      EDNCN(1;384)=EDNCN(1;384)*TDF(NPTR;384)                           0008571LINKHO 753
      EDNCN(1;384)=EDNCN(1;384)/DENO(1;384)                             0008572LINKHO 754
      EDNCN(1;384)=EDNCN(1;384)+EDN(NPTR;384)                           0008573LINKHO 755
C     IF(NCLOUD(N).GT.0) CLDFLG=.TRUE.                                  0008574LINKHO 756
      L1(1;384)=NCLOUD(NPTR;384).GT.0                                   0008575LINKHO 757
```

```
      CLDFLG(1;384)=CLDFLG(1;384).OR.L1(1;384)                        0008576LINKHO 758
C**** SET AEROSOL FLAG IF CIRRUS CLOUDS (HIGH ALBEDO)                 0008577LINKHO 759
C     IF(CLDFLG.AND.PI0.GE.1.E-4) AERFLG=.TRUE.                       0008578LINKHO 760
      L1(1;384)=PI0(1;384).GE.1.E-4                                   0008579LINKHO 761
      L1(1;384)=L1(1;384).AND.CLDFLG(1;384)                           0008580LINKHO 762
      AERFLG(1;384)=AERFLG(1;384).OR.L1(1;384)                        0008581LINKHO 763
C**** TRANSMISSION COMPUT%D DIFFERENTLY FOR 3 CASES                    0008582LINKHO 764
C     IF (CLDFLG.OR.AERFLG) GO TO 125                                 0008583LINKHO 765
      L1(1;384)=.NOT.CLDFLG(1;384).AND..NOT.AERFLG(1;384)             0008584LINKHO 766
C**** CASE 1. ATMOSPHERE HAS NO AEROSOLS OR CLOUDS THRU HERE          0008585LINKHO 767
C**** USE EXPONENTIAL INTEGRAL APPROXIMATION                          0008586LINKHO 768
      X3(1;384)=TAU(1;384)+TAUN(NKPTR;384)                            0008587LINKHO 769
      TAU(1;384) = Q8VCTRL(X3(1;384),L1(1;384);TAU(1;384))            0008588LINKHO 770
C IF (TAU.GT.15.) GO TO 124 / TDFCN=0.                                0008589LINKHO 771
      L2(1;384)=X3(1;384).GT.15.                                      0008590LINKHO 772
      L2(1;384)=L2(1;384).AND.L1(1;384)                               0008591LINKHO 773
      TDFCN(1;384)=Q8VCTRL(ZERO(1;384),L2(1;384);TDFCN(1;384))        0008592LINKHO 774
      L1(1;384)=L1(1;384).XOR.L2(1;384)                               0008593LINKHO 775
      LOUT=Q8SCNT(L1(1;384))                                          0008594LINKHO 776
      TY(1;LOUT)=Q8VCMPRS(X3(1;384),L1(1;384);TY(1;LOUT))             0008595LINKHO 777
      TY(1;LOUT)=20.*TY(1;LOUT)                                       0008596LINKHO 778
      TY(1;LOUT)=TY(1;LOUT)+1.                                        0008597LINKHO 779
      ITY(1;LOUT)=TY(1;LOUT)                                          0008598LINKHO 780
C     TDFCN=TE3(ITY)+(TY-ITY+1)*(TE3(ITY+1)-TE3(ITY))                 0008599LINKHO 781
      X2(1;LOUT)=Q8VGATHR(TE3(1;301),ITY(1;LOUT);X2(1;LOUT))          0008600LINKHO 782
      ITY(1;LOUT)=ITY(1;LOUT)+1                                       0008601LINKHO 783
      X1(1;LOUT)=Q8VGATHR(TE3(1;301),ITY(1;LOUT);X1(1;LOUT))          0008602LINKHO 784
      X1(1;LOUT)=X1(1;LOUT)-X2(1;LOUT)                                0008603LINKHO 785
      X3(1;LOUT)=TY(1;LOUT)-ITY(1;LOUT)                               0008604LINKHO 786
      X3(1;LOUT)=X3(1;LOUT)+2.                                        0008605LINKHO 787
      X3(1;LOUT)=X3(1;LOUT)*X1(1;LOUT)                                0008606LINKHO 788
      X3(1;LOUT)=X3(1;LOUT)+X2(1;LOUT)                                0008607LINKHO 789
      X1(1;384)=Q8VXPND(X3(1;LOUT),L1(1;384);X1(1;384))               0008608LINKHO 790
      TDFCN(1;384)=Q8VCTRL(X1(1;384),L1(1;384);TDFCN(1;384))          0008609LINKHO 791
C**** CASE 2. SIGNIFICANT ABSORPTION, (IF AERFLG)                      0008610LINKHO 792
C     RDNCN=REF(N)+TDF(N)*RDNCN*TDF(N)/DENO                           0008611LINKHO 793
      X3(1;384)=RDNCN(1;384)*TDF(NPTR;384)                            0008612LINKHO 794
      X3(1;384)=X3(1;384)*TDF(NPTR;384)                               0008613LINKHO 795
      X3(1;384)=X3(1;384)/DENO(1;384)                                 0008614LINKHO 796
      X3(1;384)=X3(1;384)+REF(NPTR;384)                               0008615LINKHO 797
      RDNCN(1;384)=Q8VCTRL(X3(1;384),AERFLG(1;384);RDNCN(1;384))      0008616LINKHO 798
C     TDFCN=TDFCN*TDF(N)/DENO                                         0008617LINKHO 799
      X3(1;384)=TDFCN(1;384)*TDF(NPTR;384)                            0008618LINKHO 800
      X3(1;384)=X3(1;384)/DENO(1;384)                                 0008619LINKHO 801
      TDFCN(1;384)=Q8VCTRL(X3(1;384),AERFLG(1;384);TDFCN(1;384))      0008620LINKHO 802
C130  IF (NCLOUD(N).EQ.0.OR.PI0.GE.1.E-4) GO TO 140                   0008621LINKHO 803
130   L1(1;384)=NCLOUD(NPTR;384).GT.0                                 0008622LINKHO 804
      L2(1;384)=PI0(1;384).LT.1.E-4                                   0008623LINKHO 805
      L1(1;384)=L1(1;384).AND.L2(1;384)                               0008624LINKHO 806
C**** CASE 3. HEAVY CLOUD COVER                                       0008625LINKHO 807
      TDFCN(1;384)=Q8VCTRL(ZERO(1;384),L1(1;384);TDFCN(1;384))        0008626LINKHO 808
      RDNCN(1;384)=Q8VCTRL(ZERO(1;384),L1(1;384);RDNCN(1;384))        0008627LINKHO 809
      TAU(1;384)=Q8VCTRL(ZERO(1;384),L1(1;384);TAU(1;384))            0008628LINKHO 810
140   CONTINUE                                                        0008629LINKHO 811
C**** SAVE PARTIAL SUMS                                               0008630LINKHO 812
C     EUPC(N)=EUPCN                                                   0008631LINKHO 813
      EDNC(NPTR;384)=EDNCN(1;384)                                     0008632LINKHO 814
      TDFC(NPTR;384)=TDFCN(1;384)                                     0008633LINKHO 815
      RDNC(NPTR;384)=RDNCN(1;384)                                     0008634LINKHO 816
101   CONTINUE                                                        0008635LINKHO 817
C**** ADDING GROUND LAYER                                             0008636LINKHO 818
      RUPCN(1;384)=AGRND                                              0008637LINKHO 819
C     EUPCN=(1.0-RUPCN)*BTOP(NG1)                                     0008638LINKHO 820
```

```
      EUPCN(1;384)=ONE(1;384)-RUPCN(1;384)
      EUPCN(1;384)=EUPCN(1;384)*BTOP(NG*384+1;384)
C     DENO=1.0-RUPCN*RDNCN
      X1(1;384)=RUPCN(1;384)*RDNCN(1;384)
      DENO(1;384)=ONE(1;384)-X1(1;384)
C     PEFUP=(EUPCN+EDNCN*RUPCN)/DENO
      X1(1;384)=EDNCN(1;384)*RUPCN(1;384)
      PEFUP(1;384)=EUPCN(1;384)+X1(1;384)
      PEFUP(1;384)=PEFUP(1;384)/DENO(1;384)
C     PEFDN=(EDNCN+EUPCN*RDNCN)/DENO
      X1(1;384)=EUPCN(1;384)*RDNCN(1;384)
      PEFDN(1;384)=EDNCN(1;384)+X1(1;384)
      PEFDN(1;384)=PEFDN(1;384)/DENO(1;384)
C     FLXONG=FLXDNG+CKLAM*PEFDN
      X1(1;384)=CKLAM*PEFDN(1;384)
      FLXDNG(1;384)=FLXDNG(1;384)+X1(1;384)
C     FE(NG)=FE(NG)+CKLAM*(PEFUP-PEFDN)
      X1(1;384)=PEFUP(1;384)-PEFDN(1;384)
      X1(1;384)=X1(1;384)*CKLAM
      FE((NG-1)*384+1;384)=FE((NG-1)*384+1;384)+X1(1;384)
C****-
C**** FORM BOTTOM COMPOSITE LAYER (ADDITION)
C****
      DO 118 N=2,NG
      M=NG-N+1
      MPTR=(M-1)*384+1
      X1(1;384)=RUPCN(1;384)*REF(MPTR;384)
      DENO(1;384)=ONE(1;384)-X1(1;384)
C     EUPCN=EUP(M)+(EUPCN+EDN(M)*RUPCN)*TDF(M)/DENO
      X1(1;384)=EDN(MPTR;384)*RUPCN(1;384)
      X1(1;384)=X1(1;384)+EUPCN(1;384)
      X1(1;384)=X1(1;384)*TDF(MPTR;384)
      X1(1;384)=X1(1;384)/DENO(1;384)
      EUPCN(1;384)=EUP(MPTR;384)+X1(1;384)
      IF (M.EQ.1) GO TO 119
      L=M-1
      LPTR=MPTR-384
C     RUPCN=REF(M)+TDF(M)*TDF(M)*RUPCN/DENO
      X1(1;384)=TDF(MPTR;384)*TDF(MPTR;384)
      X1(1;384)=X1(1;384)*RUPCN(1;384)
      X1(1;384)=X1(1;384)/DENO(1;384)
      RUPCN(1;384)=REF(MPTR;384)+X1(1;384)
C     DENO=1.0-RDNC(L)*RUPCN
      X1(1;384)=RDNC(LPTR;384)*RUPCN(1;384)
      DENO(1;384)=ONE(1;384)-X1(1;384)
C     PEFUP=(EURCN+FDNC(L)*RUPCN)/DENO
      X1(1;384)=EDNC(LPTR;384)*RUPCN(1;384)
      PEFUP(1;384)=EUPCN(1;384)+X1(1;384)
      PEFUP(1;384)=PEFUP(1;384)/DENO(1;384)
C     PEFDN=(EDNC(L)+EUPCN*RDNC(L))/DENO
      X1(1;384)=EUPCN(1;384)*RDNC(LPTR;384)
      PEFDN(1;384)=EDNC(LPTR;384)+X1(1;384)
      PEFDN(1;384)=PEFDN(1;384)/DENO(1;384)
      GO TO 120
119   PEFUP(1;384)=EUPCN(1;384)
      PEFDN(1;384)=0.
C****
C120  FE(M)=FE(M)+CKLAM*(PEFUP-PEFDN)
120   X1(1;384)=PEFUP(1;384)-PEFDN(1;384)
      X1(1;384)=CKLAM*X1(1;384)
      FE(MPTR;384)=FE(MPTR;384)+X1(1;384)
C****
118   CONTINUE
100   CONTINUE
200   CONTINUE
C**** SAVE STRATOSPHERIC FLUXES
      RESTR(1;1152)=FE(1;1152)
      RE(1;3840)=FE(1153;3840)
      RETURN
      END
```

APPENDIX C

SINGLE, SIMPLE LOOPS OF SPECTRAL MODEL

```
      SUBROUTINE  XIGENR
      DOUBLE PRECISION VDERIV,XDERIV,W,DXO3DT
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT,YRLAG,TIME
      COMMON /QJBLK/ NZJ,L103
      COMMON /DERIV/ VDERIV(2366),XDERIV(2366),W(2366)
      COMMON /GENER/ DXO3DT(2366)
      IL=(L103-1)*NVREAL+1
      IH=NZJ*NVREAL
      DO 200  I=IL,IH
  200 XDERIV(I)=XDERIV(I)+DXO3DT(I)
      RETURN
      END
```

```
      SUBROUTINE OXTO03(I1,I2)
      DOUBLE PRECISION P,Z,Z1,T,Z2,X3
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366),X3(2366)
      COMMON/O3OX/ O3XFAC(2400),O3XCON(2400)
      COMMON/SPECIE/X3GRD(6240)
      COMMON/FTCST/NLON,NLAT,NGRID
      COMMON/CONSTS/L(13),NVREAL
      DIMENSION DATAIM(2400)
      NLEV=I2-I1+1
      ILSPC=(I1-1)*NVREAL+1
      ILGRD=(I1-1)*NGRID+1
      CALL SPCGD1(X3(ILSPC),X3GRD(ILGRD),DATAIM,NLEV)
      N=I2*NGRID
      DO 100 J=ILGRD,N
  100 X3GRD(J)=(X3GRD(J)-O3XCON(J))/O3XFAC(J)
      CALL GDSP 1(X3(ILSPC),X3GRD(ILGRD),DATAIM,NLEV)
      RETURN
      END
```

APPENDIX D

PARTIALLY RECODED SUBROUTINES OF SPECTRAL MODEL

```
      SUBROUTINE CORFOR
      DOUBLE PRECISION P,Z,Z1,T,Z2,CF,XL
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /DERIV/ CF(2366)
      DIMENSION DUM(43)
      DATA NW /0/
      NZ1=NVZON+1
      DO 300  JJ=NZ1,NVECT
      J=2*JJ+NVREAL*(ILEV1-2)-NZ1
      XL=LV(JJ)
      DO 200  I=ILEV1,ILEV2
      J=J+NVREAL
      J1=J+1
      CF(J)=XL*P(J1)
      CF(J1)=-XL*P(J)
  200 CONTINUE
  300 CONTINUE
      IF (NW.EQ.0) RETURN
      ILEV=(ILEV2-ILEV1)/2+1
      JMP=NVREAL*(ILEV-1)
      WRITE (6,1000) ILEV
 1000 FORMAT (1H0,10X,'TEST CORFOR ENERGY CONSERVATION FOR LEVEL ',I3/)
      DO 510  J=1,NVZON
      JL=J+JMP
  510 DUM(J)=-.5*P(JL)*CF(JL)
      DO 530  J=NZ1,NVECT
      JR=2*J-NZ1+JMP
      JI=JR+1
  530 DUM(J)=-P(JR)*CF(JR)-P(JI)*CF(JI)
      SUM=0.
      DO 540  J=1,NVECT
  540 SUM=SUM+DUM(J)
      WRITE (6,1010)
 1010 FORMAT (1H0,10X,'PSI(I),CF(I),DKE(I) ='/)
      DO 550  I=1,NVZON
      J=I+JMP
      WRITE (6,1015) I,P(J),CF(J),DUM(I)
 1015 FORMAT (5X,I5,E15.6,15X,E15.6,15X,E15.6)
  550 CONTINUE
      K=NVZON
      DO 600  I=NZ1,NVREAL,2
      II=I+JMP
      J=II+1
      K=K+1
      WRITE (6,1020) I,P(II),P(J),CF(II),CF(J),DUM(K)
 1020 FORMAT (5X,I5,5E15.6)
  600 CONTINUE
      WRITE (6,1030) SUM
 1030 FORMAT (1H0,10X,'TOTAL DKE = ',E15.6)
      RETURN
      END
```

```
      SUBROUTINE VCORFOR
         .
         .
         .
      J=NVZON-1
      DO 300 JJ=NZ1,NVECT
      J=J+2
      XL=LV(JJ)
      DO 200 I=ILEV1,ILEV2
      CF(J,I)=XL*P(J+1,I)
      CF(J+1,I)=-XL*P(J,I)
  200 CONTINUE
  300 CONTINUE
      RETURN

      ENTRY CORFOR0
      NZ1=NVZON+1
      RETURN
      END
```

```
      SUBROUTINE FRICTN
      DOUBLE PRECISION P,Z,FJ,CF
      COMMON P(2366),Z(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /DERIV/ CF(2366)
      COMMON /BARBLK/ TBAR(26),SIGMA(26),XIBAR(26),DIFFM(26),DIFFX(26)
      DIMENSION FJ(26)
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      IL1P1=ILEV1+1
      IL2P1=ILEV2+1
      FJ(1)=0.D0
      DO 300  J=1,NVREAL
      JJ=(ILEV1-1)*NVREAL+J
      J1=JJ
      DO 100  I=IL1P1,ILEV2
      J2=J1
      J1=J1+NVREAL
100   FJ(I)=DIFFM(I)*(Z(J1)-Z(J2))
      FJ(IL2P1)=-DIFFM(IL2P1)*Z(J1)
      JJ=JJ-NVREAL
      DO 200  I=ILEV1,ILEV2
      JJ=JJ+NVREAL
200   CF(JJ)=CF(JJ)+R1*FJ(I+1)-R2*FJ(I)
300   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VFRICTN
         .
         .
         .
      FJ(1)=0.
      DO 300 J=1,NVREAL
      DO 100 I=IL1P1,ILEV2
      FJ(I)=DIFFM(I)*(Z(J,I)-Z(J,I-1))
 100  CONTINUE
      FJ(IL2P1)=-DIFFM(IL2P1)*Z(J,ILEV2)
      DO 200 I=ILEV1,ILEV2
      CF(J,I)=CF(J,I)+R1*FJ(I+1)-R2*FJ(I)
 200  CONTINUE
 300  CONTINUE
      RETURN

      ENTRY FRICTNO
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      IL1P1=ILEV1+1
      IL2P1=ILEV2+1
      RETURN
      END
```

```
      SUBROUTINE MJAB (P,T,DER)
      DOUBLE PRECISION P,T,A,C,F1R,F1I,CIND,FR,FI,F1A,F1B,F1C,F1D
      DOUBLE PRECISION DER
      COMMON /COFBLK/ C(3800),IS(1500)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON/PKBLK/NS,N1,N2,N3,N4,LS,L1,L2,L3,L4
      COMMON /WORKBK/ A(2366)
      DIMENSION P(1),T(1),DER(1)
      DIMENSION FR(26),FI(26)
      CALL DZERO (A)
      INDEX=0
      KLOW=(ILEV1-1)*KINT
      INSZ1=INSZ+1
      DO 400 K=1,INSZ
      LS=IS(K)
      CALL UPACK
      NAI=N1+N1-NR-1
      NAR=NAI-1
      NBI=N2+N2-NR-1
      NBR=NBI-1
      K1=KLOW
      DO 5  J=ILEV1,ILEV2
      NAR=NAR+K1
      NAI=NAI+K1
      NBR=NBR+K1
      NBI=NBI+K1
      FR(J)= P(NAR)*T(NBI)-P(NAI)*T(NBR)+P(NBR)*T(NAI)-P(NBI)*T(NAR)
      K1=KINT
    5 CONTINUE
      DO 100 I=N3,N4,2
      INDEX=INDEX+1
      CIND=C(INDEX)
      K1=KLOW
      DO 75  J=ILEV1,ILEV2
      A(I+K1)=A(I+K1)+FR(J)*CIND
      K1=K1+KINT
   75 CONTINUE
  100 CONTINUE
  400 CONTINUE
      DO 500 K=INSZ1,INS
      LS=IS(K)
      CALL UPACK
      K1=KLOW
      IF (N1.LE.NR+1) GO TO 40
      IF (N2.LE.NR+1) GO TO 50
      NAI=N1+N1-NR-1
      NAR=NAI-1
      NBI=N2+N2-NR-1
      NBR=NBI-1
      DO 35  J=ILEV1,ILEV2
      NAR=NAR+K1
      NAI=NAI+K1
      NBR=NBR+K1
      NBI=NBI+K1
      F1A=P(NAR)*T(NBI)-P(NBI)*T(NAR)
      F1B=P(NAI)*T(NBR)-P(NBR)*T(NAI)
      F1C=P(NAR)*T(NBR)-P(NBR)*T(NAR)
      F1D=P(NAI)*T(NBI)-P(NBI)*T(NAI)
```

```
      IF (NS-1) 10,20,30
   10 F1R=-F1A-F1B
      F1I=F1C-F1D
      GO TO 33
   20 F1R=-F1A+F1B
      F1I=F1C+F1D
      GO TO 33
   30 F1R=F1A-F1B
      F1I=F1C+F1D
      GO TO 33
   33 CONTINUE
      FR(J)=F1R
      FI(J)=F1I
      K1=KINT
   35 CONTINUE
      GO TO 160
   40 NBI=N2+N2-NR-1
      NBR=NBI-1
      NAI=N1
      NAR=N1
      GO TO 60
   50 NAR=N1+N1-NR-1
      NAI=NAR-1
      NBR=N2
      NBI=N2
      GO TO 60
   60 CONTINUE
      DO 150  J=ILEV1,ILEV2
      NAR=NAR+K1
      NAI=NAI+K1
      NBR=NBR+K1
      NBI=NBI+K1
      F1R=-P(NAR)*T(NBI)+P(NBI)*T(NAR)
      F1I=P(NAI)*T(NBR)-P(NBR)*T(NAI)
      GO TO 80
   80 CONTINUE
      FR(J)=F1R
      FI(J)=F1I
      K1=KINT
  150 CONTINUE
  160 CONTINUE
      NGI=N3+N3-NR-1
      DO 200 I=N3,N4,2
      NGR=NGI-1
      INDEX=INDEX+1
      CIND=C(INDEX)
      K1=KLOW
      DO 175  J=ILEV1,ILEV2
      A(NGR+K1)=A(NGR+K1)+FR(J)*CIND
      A(NGI+K1)=A(NGI+K1)+FI(J)*CIND
      K1=K1+KINT
  175 CONTINUE
      NGI=NGI+4
  200 CONTINUE
  500 CONTINUE
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 600  I=IL,IH
  600 DER(I)=DER(I)+A(I)
      RETURN
      END
```

```
      SUBROUTINE VMJAB(P,T,DER)
         .
         .
         .
      CALL DZERO(A)

      INDEX=0
      DO 400 K=1,INSZ
      LS=IS(K)

      CALL UPACK

      DO 5 J=ILEV1,ILEV2
      FR(J)=P(NAR,J)*T(NBI,J)-P(NAI,J)*T(NBR,J)
     1      +P(NBR,J)*T(NAI,J)-P(NBI,J)*T(NAR,J)
    5 CONTINUE
      DO 100 I=N3,N4,2
      INDEX=INDEX+1
      DO 75 J=ILEV1,ILEV2
      A(I,J)=A(I,J)+FR(J)*C(INDEX)
   75 CONTINUE
  100 CONTINUE
  400 CONTINUE
      DO 500 K=INSZ1,INS
      LS=IS(K)

      CALL UPACK

      DO 35 J=ILEV1,ILEV2
      F1A(J)=P(NAR,J)*T(NBI,J)-P(NBI,J)*T(NAR,J)
      F1B(J)=P(NAI,J)*T(NBR,J)-P(NBR,J)*T(NAI,J)
      F1C(J)=P(NAR,J)*T(NBR,J)-P(NBR,J)*T(NAR,J)
      F1D(J)=P(NAI,J)*T(NBI,J)-P(NBI,J)*T(NAI,J)
   10 CONTINUE
      FR(J)=-F1A(J)-F1B(J)
      FI(J)=F1C(J)-F1D(J)
   35 CONTINUE
      NGI=NGI0
      DO 200 I=N3,N4,2
      NGR=NGI-1
      INDEX=INDEX+1
      DO 175 J=ILEV1,ILEV2
      A(NGR,J)=A(NGR,J)+FR(J)*C(INDEX)
      A(NGI,J)=A(NGI,J)+FI(J)*C(INDEX)
  175 CONTINUE
      NGI=NGI+4
  200 CONTINUE
  500 CONTINUE
      DO 600 I=1,NVREAL
      DO 600 J=ILEV1,ILEV2
      DER(I,J)=DER(I,J)+A(I,J)
  600 CONTINUE
      RETURN

      ENTRY MJAB0
      INSZ1=INSZ+1
      NAI=N1+N1-NR-1
      NAR=NAI-1
      NBI=N2+N2-NR-1
      NBR=NBI-1
      NGI0=N3+N3-NR-1
      RETURN
      END
```

```
      SUBROUTINE RGAMMA (N)
      DOUBLE PRECISION P,ZETA,Z1,T,Z2,DG,EG,DGCG,EGCG
      DOUBLE PRECISION EVECT,XLIVET,EVAL,VDERIV,TDERIV,R,AG
      DOUBLE PRECISION A
      COMMON P(2366),ZETA(2366),Z1(2366),T(2366),Z2(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /DEBLK/ DG(43),EG(43),DGCG(43),EGCG(43)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /DERIV/ VDERIV(2366),TDERIV(2366),W(2366)
      COMMON /WORKBK/ R(2366),AG(2366)
      DATA NW /0/
      A=-(N-1.D0)/(NCYC*DT)
      RVM1=RV-1.
      IH=ILEV2*NVREAL
      IL=(ILEV1-1)*NVREAL+1
      IF (N.GT.1) GO TO 75
      DO 50  I=IL,IH
      J=I-NVREAL
      AG(I)=VDERIV(J)-VDERIV(I)
      R(I)=-TDERIV(I)*DZ
 50   CONTINUE
      GO TO 110
 75   CONTINUE
      DO 100  I=IL,IH
      J=I-NVREAL
      AG(I)=VDERIV(J)-VDERIV(I)+A*(Z1(J)-Z1(I))
      R(I)=-(TDERIV(I)+A*Z2(I))*DZ
100   CONTINUE
110   CONTINUE
      NVZ1=NVZON+1
      DO 200  JJ=2,NVZON
      JSV=JJ+NVREAL*(ILEV1-2)
      IF (JJ.EQ.2) GO TO 150
      F=-DGCG(JJ)/CG(JJ)
      J=JSV
      DO 125  I=ILEV1,ILEV2
      J=J+NVREAL
      J1=J-1
      R(J)=R(J)+F*AG(J1)
125   CONTINUE
150   CONTINUE
      IF (JJ.EQ.NVZON) GO TO 200
      J=JSV
      F=EGCG(JJ)/CG(JJ)
      DO 175  I=ILEV1,ILEV2
      J=J+NVREAL
      J1=J+1
      R(J)=R(J)+F*AG(J1)
175   CONTINUE
200   CONTINUE
```

```
      JJ=NVZON
      DO 500  L=1,LR
      NC=NCOMP(L+1)
      IF (NC.LE.0) GO TO 500
      DO 450  NN=1,NC
      JJ=JJ+1
      JSV=2*JJ+NVREAL*(ILEV1-2)-NVZON-1
      IF (NN.EQ.1) GO TO 300
      F=-DGCG(JJ)/CG(JJ)
      JR=JSV
      DO 250  I=ILEV1,ILEV2
      JR=JR+NVREAL
      JI=JR+1
      J1R=JR-2
      J1I=J1R+1
      R(JR)=R(JR)+F*AG(J1R)
      R(JI)=R(JI)+F*AG(J1I)
  250 CONTINUE
  300 CONTINUE
      IF (NN.EQ.NC) GO TO 500
      JR=JSV
      F=EGCG(JJ)/CG(JJ)
      DO 400  I=ILEV1,ILEV2
      JR=JR+NVREAL
      JI=JR+1
      J1R=JR+2
      J1I=J1R+1
      R(JR)=R(JR)+F*AG(J1R)
      R(JI)=R(JI)+F*AG(J1I)
  400 CONTINUE
  450 CONTINUE
  500 CONTINUE
      DO 600  I=IL,IH
      R(I)=RVM1*R(I)
  600 CONTINUE
      IF (NW.EQ.0) RETURN
      WRITE (6,1000) ILEV1,ILEV2
 1000 FORMAT (1H0,10X,'I , R(',I2,') , R(',I2,') ='/)
      J1=NVREAL*(ILEV1-1)
      J3=NVREAL*(ILEV2-1)
      DO 700  I=1,NVZON
      J1=J1+1
      J3=J3+1
      WRITE (6,1010) I, R(J1),R(J3)
 1010 FORMAT (1X,I10,D20.10,20X,D20.10)
  700 CONTINUE
      NZ1=NVZON+1
      DO 800  I=NZ1,NVREAL,2
      J1=J1+1
      J2=J1+1
      J3=J3+1
      J4=J3+1
      WRITE (6,1020) I,R(J1),R(J2),R(J3),R(J4)
 1020 FORMAT (1X,I10,4D20.10)
      J1=J1+1
      J3=J3+1
  800 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VRGAMMA(N)
         .
         .
         .
      IF (N.GT.1) GO TO 75
      DO 50 J=1,NVREAL
      DO 50 I=ILEV1,ILEV2
      AG(J,I)=VDERIV(J,I-1)-VDERIV(J,I)
      R(J,I)=-TDERIV(J,I)*DZ
 50   CONTINUE
      GO TO 110
 75   CONTINUE
      DO 100 J=1,NVREAL
      DO 100 I=ILEV1,ILEV2
      AG(J,I)=VDERIV(J,I-1)-VDERIV(J,I)+A(N)*(Z1(J,I-1)-Z1(J,I))
      R(J,I)=-(TDERIV(J,I)+A(N)*Z2(J,I))*DZ
100   CONTINUE
110   CONTINUE
      DO 200 JJ=2,NVZON
      DO 165 I=ILEV1,ILEV2
      R(JJ,I)=R(JJ,I)+F(JJ)*AG(JJ-1,I)
165   CONTINUE
200   CONTINUE
      DO 500 JJ=NZ1,NVECT
      J=2*JJ-NZ1
      DO 350 I=ILEV1,ILEV2
      R(J,I)=R(J,I)+F(JJ)*AG(J-2,I)
      R(J+1,I)=R(J+1,I)+F(JJ)*AG(J-1,I)
350   CONTINUE
500   CNMSIMTE
      DO 600 I=IL,IH
      R(I)=R(I)*RVM1
600   CONTINUE
      RETURN

      ENTRY RGAMMA0
      NZ1=NVZON+1
      F(2)=EGCG(2)/CG(2)
      DO 1 JJ=3,NVZON-1
      F(JJ)=(EGCG(JJ)-DGCG(JJ))/CG(JJ)
  1   CONTINUE
      F(NVZON)=-DGCG(NVZON)/CG(NVZON)
      JJ=NVZON
      DO 3 L=1,LR
      JJ=JJ+1
      F(JJ)=EGCG(JJ)/CG(JJ)
      DO 2 NN=2,NC-1
      JJ=JJ+1
      F(JJ)=(EGCG(JJ)-DGCG(JJ))/CG(JJ)
  2   CONTINUE
      JJ=JJ+1
      F(JJ)=-DGCG(JJ)/CG(JJ)
  3   CONTINUE
      DO 4 I=1,NCYC
      A(I)=(1.-I)/(NCYC*DT)
  4   CONTINUE
      RVM1=RV-1.
      IH=ILEV2*NVREAL
      IL=(ILEV1-1)*NVREAL+1
      NC=NCOMP(2)
      RETURN
      END
```

```
      SUBROUTINE WFIELD
      DOUBLE PRECISION DG,EG,DGCG,EGCG,W,WTERM,E1,E2,F1,F2,F3,F4
      DOUBLE PRECISION VDERIV,TDERIV
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /DEBLK/ DG(43),EG(43),DGCG(43),EGCG(43)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /DERIV/ VDERIV(2366),TDERIV(2366),W(2366)
      COMMON /WORKBK/ WTERM(2366)
      CALL DZERO (WTERM)
      E2=1.D0/(RV-1.D0)
      E1=RV*E2
      NVZ1=NVZON+1
      DO 200  JJ=2,NVZON
      JSV=JJ+NVREAL*(ILEV1-2)
      IF (JJ.EQ.2) GO TO 100
      F1=E2*DGCG(JJ)
      F2=E1*DGCG(JJ)
      J=JSV
      DO 50  I=ILEV1,ILEV2
      J=J+NVREAL
      J1=J-1
      J2=J1+NVREAL
      WTERM(J)=F1*W(J1)-F2*W(J2)
   50 CONTINUE
  100 CONTINUE
      IF (JJ.EQ.NVZON) GO TO 200
      J=JSV
      F1=E2*EGCG(JJ)
      F2=E1*EGCG(JJ)
      DO 150  I=ILEV1,ILEV2
      J=J+NVREAL
      J1=J+1
      J2=J1+NVREAL
      WTERM(J)=WTERM(J)-F1*W(J1)+F2*W(J2)
  150 CONTINUE
  200 CONTINUE
      JJ=NVZON
      DO 500  L=1,LR
      N=NCOMP(L+1)
      IF (N.LE.0) GO TO 500
      DO 450  NN=1,N
      JJ=JJ+1
      JSV=2*JJ+NVREAL*(ILEV1-2)-NVZON-1
      IF (NN.EQ.1) GO TO 300
      F1=E2*DGCG(JJ)
      F2=E1*DGCG(JJ)
      JR=JSV
      DO 250  I=ILEV1,ILEV2
      JR=JR+NVREAL
      JI=JR+1
      J1R=JR-2
      J1I=J1R+1
      J2R=J1R+NVREAL
      J2I=J2R+1
      WTERM(JR)=F1*W(J1R)-F2*W(J2R)
      WTERM(JI)=F1*W(J1I)-F2*W(J2I)
  250 CONTINUE
  300 CONTINUE
```

```
      IF (NN.EQ.N) GO TO 500
      JR=JSV
      F1=E2*EGCG(JJ)
      F2=E1*EGCG(JJ)
      DO 400  I=ILEV1,ILEV2
      JR=JR+NVREAL
      JI=JR+1
      J1R=JR+2
      J1I=J1R+1
      J2R=J1R+NVREAL
      J2I=J2R+1
      WTERM(JR)=WTERM(JR)-F1*W(J1R)+F2*W(J2R)
      WTERM(JI)=WTERM(JI)-F1*W(J1I)+F2*W(J2I)
  400 CONTINUE
  450 CONTINUE
  500 CONTINUE
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 600  I=IL,IH
  600 VDERIV(I)=VDERIV(I)+WTERM(I)
      RETURN
      END
```

```
      SUBROUTINE VWFIELD
        .
        .
        .
      DO 200 JJ=2,NVZON
      DO 165 I=ILEV1,ILEV2
      WTERM(JJ,I)=F1(JJ)*W(JJ-1,I)-F2(JJ)*W(JJ-1,I+1)
  165 CONTINUE
  200 CONTINUE
      DO 500 JJ=NVZ1,NVECT
      J=2*JJ-NVZ1
      DO 360 I=ILEV1,ILEV2
      WTERM(J,I)=F1(JJ)*W(J-2,I)-F2(JJ)*W(J-2,I+1)
      WTERM(J+1,I)=F1(JJ)*W(J-1,I)-F2(JJ)*W(J-1,I+1)
  360 CONTINUE
  500 CONTINUE
      DO 600 I=IL,IH
      VDERIV(I)=VDERIV(I)+WTERM(I)
  600 CONTINUE
      RETURN

      ENTRY WFIELD0
      NVZ1=NVZON+1
      E2=1./(RV-1.)
      E1=RV*E2
      F1(2)=-E2*EGCG(2)
      F2(2)=-E1*EGCG(2)
      DO 1 J=3,NVZON-1
      F1(J)=E2*(DGCG(J)-EGCG(J))
      F2(J)=E1*(DGCG(J)-EGCG(J))
    1 CONTINUE
      F1(NVZON)=E2*DGCG(NVZON)
      F2(NVZON)=E1*DGCG(NVZON)
      JJ=NVZON
      DO 3 L=1,LR
      JJ=JJ+1
      F1(JJ)=-E2*EGCG(JJ)
      F2(JJ)=-E1*EGCG(JJ)
      DO 2 NN=2,N-1
      JJ=JJ+1
      F1(JJ)=E2*(DGCG(JJ)-EGCG(JJ))
      F2(JJ)=E1*(DGCG(JJ)-EGCG(JJ))
    2 CONTINUE
      JJ=JJ+1
      F1(JJ)=E2*DGCG(JJ)
      F2(JJ)=E1*DGCG(JJ)
    3 CONTINUE
      IL=(ILEV1-1)*NVREAL
      IH=ILEV2*NVREAL
      RETURN
      END
```

```
      SUBROUTINE STABLE
      DOUBLE PRECISION W,STABW,F,VDERIV,TDERIV
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1     NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /BARBLK/ TBAR(26),SIGMA(26),XIBAR(26)
      COMMON /DERIV/ VDERIV(2366),TDERIV(2366),W(2366)
      COMMON /WORKBK/ STABW(2366)
      DO 200  I=ILEV1,ILEV2
      JJ=(I-1)*NVREAL
      F=-SIGMA(I)
      DO 100  J=1,NVREAL
      K=JJ+J
      STABW(K)=F*W(K)
  100 CONTINUE
  200 CONTINUE
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 300  I=IL,IH
  300 TDERIV(I)=TDERIV(I)+STABW(I)
      RETURN
      END
```

```
      SUBROUTINE VSTABLE
        .
        .
        .
      DO 300 I=ILEV1,ILEV2
      F=-SIGMA(I)
      DO 300 J=1,NVREAL
      STABW(J,I)=F*W(J,I)
      TDERIV(J,I)=TDERIV(J,I)+STABW(J,I)
  300 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DIFFXI
      DOUBLE PRECISION P,Z,Z1,T,Z2,X3,Z3
      DOUBLE PRECISION VDERIV,XDERIV,W,GJ
      DOUBLE PRECISION DUM
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366),X3(2366),Z3(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /BARBLK/ TBAR(26),SIGMA(26),XIBAR(26),DIFFM(26),DIFFX(26)
      COMMON /DERIV/ VDERIV(2366),XDERIV(2366),W(2366)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /QJBLK/ NZJ,L103
      DIMENSION GJ(26)
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      L103M1=L103-1
      IF (L103M1.LE.0) L103M1=1
      DO 100  I=1,L103
100   GJ(I)=0.D0
      DO 300  J=1,NVREAL
      JJ=(L103M1-1)*NVREAL+J
      J2=JJ
      DO 200  I=L103M1,ILEV2
      J1=J2+NVREAL
      GJ(I)=DIFFX(I)*(X3(J1)-X3(J2))
      J2=J1
200   CONTINUE
      JJ=(L103-2)*NVREAL+J
      DO 250  I=L103,ILEV2
      JJ=JJ+NVREAL
      DUM=R1*GJ(I)-R2*GJ(I-1)
      XDERIV(JJ)=XDERIV(JJ)+DUM
250   CONTINUE
300   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VDIFFXI
        .
        .
        .
      DO 100 I=1,L103
      GJ(I)=0.
100   CONTINUE
      DO 300 J=1,NVREAL
      DO 200 I=L103M1,ILEV2
      GJ(I)=DIFFX(I)*(X3(J,I+1)-X3(J,I))
200   CONTINUE
      DO 250 I=L103,ILEV2
      XDERIV(J,I)=XDERIV(J,I)+R1*GJ(I)-R2*GJ(I-1)
250   CONTINUE
300   CONTINUE
      RETURN

      ENTRY DIFFXI0
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      L103M1=L103-1
      IF (L103M1.LE.0) L103M1=1
      RETURN
      END
```

```
      SUBROUTINE WADVXI
      DOUBLE PRECISION P,Z,Z1,T,Z2,X3,Z3
      DOUBLE PRECISION VDERIV,XDERIV,W,WDX3DZ,WXJBAR
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366),X3(2366),Z3(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /DERIV/ VDERIV(2366),XDERIV(2366),W(2366)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /QJBLK/ NZJ,L103
      COMMON /WORKBK/ WDX3DZ(2366)
      DIMENSION WXJBAR(26)
      DZ2=2.*DZ
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      DO 200  I=L103,ILEV2
      JJ=(I-1)*NVREAL
      IL=JJ-NVREAL+1
      IH=JJ+NVREAL+1
      F=(X3(IH)-X3(IL))/DZ2
      DO 100  J=2,NVREAL
      K=JJ+J
      WDX3DZ(K)=F*W(K)
  100 CONTINUE
  200 CONTINUE
      NZ1=NVZON+1
      IL2P1=ILEV2+1
      DO 250  I=1,IL2P1
  250 WXJBAR(I)=0.D0
      IL2M1=ILEV2-1
      DO 400  I=L103,ILEV2
      JJ=(I-1)*NVREAL
      DO 300  J=2,NVZON
      K=JJ+J
  300 WXJBAR(I)=WXJBAR(I)+W(K)*X3(K)
      DO 350  J=NZ1,NVREAL,2
      K=JJ+J
      K1=K+1
      WXJBAR(I)=WXJBAR(I)+2.*(W(K)*X3(K)+W(K1)*X3(K1))
  350 CONTINUE
  400 CONTINUE
      IL2P1=ILEV2+1
      CALL XDERSP (WXJBAR,L103,IL2P1)
      DO 500  I=L103,ILEV2
      JJ=(I-1)*NVREAL+1
      WDX3DZ(JJ)=WXJBAR(I)
  500 CONTINUE
      IL=(L103-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 600  I=IL,IH
  600 XDERIV(I)=XDERIV(I)+WDX3DZ(I)
      RETURN
      END
```

```
      SUBROUTINE VWADVXI
         .
         .
         .
      DO 200 I=L103,ILEV2
      F=(X(1,I+1)-X(1,I-1))*EZ2
      DO 100 J=2,NVREAL
      WDX3DZ(J,I)=F*W(J,I)
100   CONTINUE
200   CONTINUE
      DO 250 I=1,IL2P1
      WXJBAR(I)=0.
250   CONTINUE
      DO 400 I=L103,ILEV2
      DO 300 J=2,NVZON
      WXJBAR(I)=WXJBAR(I)+W(J,I)*X3(J,I)
300   CONTINUE
      DO 350 J=NZ1,NVREAL,2
      WXJBAR(I)=WXJBAR(I)+2.*(W(J,I)*X3(J,I)+W(J+1,I)*X3(J+1,I))
350   CONTINUE
400   CONTINUE

      CALL XDERSP(WXJBAR,L103,IL2P1)

      DO 500 I=L103,ILEV2
      WDX3DZ(1,I)=WXJBAR(I)
500   CONTINUE
      DO 600 I=IL,IH
      XDERIV(I)=XDERIV(I)+WDX3DZ(I)
600   CONTINUE
      RETURN

      ENTRY WADVXI0
      DZ2=2.*DZ
      EZ2=1./DZ2
      R2=RV-1.
      R2=1./R2
      R1=RV*R2
      NZ1=NVZON+1
      IL2P1=ILEV2+1
      IL2M1=ILEV2-1
      IL=(L103-1)*NVREAL+1
      IH=ILEV2*NVREAL
      RETURN
      END
```

```
      SUBROUTINE O3SURF
      DOUBLE PRECISION P,Z,Z1,T,Z2,X3,Z3
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366),X3(2366),Z3(2366)
      COMMON / ONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /BARBLK/ TBAR(26),SIGMA(26),XIBAR(26),DIFFM(26),DIFFX(26)
      DSURF=DIFFX(NVERT)
      JJ=(NVERT-1)*NVREAL
      II=JJ-NVREAL
      DO 100  I=1,NVREAL
      JJ=JJ+1
      II=II+1
100   X3(JJ)=DSURF*X3(II)
      RETURN
      END
```

```
      SUBROUTINE VO3SURF
        .
        .
        .
      DO 100 I=1,NVREAL
      X3(I,NVERT)=DSURF*X(I,NVERT-1)
100   CONTINUE
      RETURN

      ENTRY O3SURFO
      DSURF=DIFFX(NVERT)
      RETURN
      END
```

```
      SUBROUTINE STREAM (N)
      DOUBLE PRECISION Z,P,X
      COMMON P(2366),Z(2366)
      COMMON / GBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      DATA NW /0/
      NVZ1=NVZON+1
      DO 200  JJ=2,NVZON
      X=-1.D0/CG(JJ)
      J=JJ-NVREAL
      DO 100  I=ILEV1,ILEV2
      J=J+NVREAL
      P(J)=X*Z(J)
100   CONTINUE
```

```
  200 CONTINUE
      DO 400  JJ=NVZ1,NVECT
      X=-1.D0/CG(JJ)
      J=2*JJ+NVREAL*(ILEV1-2)-NVZON-1
      DO 300  I=ILEV1,ILEV2
      J=J+NVREAL
      P(J)=X*Z(J)
      J1=J+1
      P(J1)=X*Z(J1)
  300 CONTINUE
  400 CONTINUE
      NW=0
      IF (N.EQ.1) NW=1
      IF (NW.EQ.0) RETURN
      DO 600  J=1,2
      GO TO (450,460),J
  450 WRITE (6,1000) ILEV1
 1000 FORMAT (1H0,10X,'LEVEL ',I3/11X,'I , VORT(I) , PSI(I) ='/)
      JJ=NVREAL*(ILEV1-1)
      GO TO 470
  460 WRITE (6,1000) ILEV2
      JJ=NVREAL*(ILEV2-1)
      GO TO 470
  470 CONTINUE
      DO 500  I=1,NVZON
      IR=I+JJ
      WRITE (6,1010) I,Z(IR),P(IR)
 1010 FORMAT (1X,I10,D20.10,20X,D20.10)
  500 CONTINUE
      DO 550  I=NVZ1,NVREAL,2
      IR=I+JJ
      II=IR+1
      WRITE (6,1020) I,Z(IR),Z(II),P(IR),P(II)
 1020 FORMAT (1X,I10,4D20.10)
  550 CONTINUE
  600 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VSTREAM
        .
        .
        .
      DO 401 J=2,NVREAL
      DO 401 I=ILEV1,ILEV2
      P(J,I)=X(J)*Z(J,I)
  401 CONTINUE
      RETURN

      ENTRY STREAM0
      DO 200 JJ=2,NVZON
      X(JJ)=-1./CG(JJ)
  200 CONTINUE
      NVZ1=NVZON+1
      DO 400 JJ=NVZ1,NVECT
      J=2*JJ-NVZ1
      X(J)=-1./CG(JJ)
      X(J+1)=-1./CG(JJ)
  400 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE THWIND
      DOUBLE PRECISION P,ZETA,Z1,T,Z2,DG,EG
      COMMON P(2366),ZETA(2366),Z1(2366),T(2366),Z2(2366)
      COMMON / ONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /DEBLK/ DG(43),EG(43)
      COMMON /VRTBLK/ ZVAL(26),PVAL(26),VWT(26),DZ,RV
      COMMON /BARBLK/ TBAR(26),SIGMA(26),XIBAR(26)
      NW=0
      IF (ILEV1.EQ.0) NW=1
      ILEV1=2
      ILEV2=NVERT-1
C
C     MEAN T COMPUTATION.
C
      J=(ILEV1-2)*NVREAL+1
      DO 25  I=ILEV1,ILEV2
      J=J+NVREAL
   25 T(J)=TBAR(I)
C
C     T FROM THERMAL WIND COMPONENTS.
C
      N=NCOMP(1)
      JB=1
      IF (N.LT.2) GO TO 250
      DO 200  JJ=2,N
      JB=JB+1
      J=JB+(ILEV1-2)*NVREAL
      DUM=DZ*CG(JJ)
      X1=DG(JJ)/DUM
      X2=EG(JJ)/DUM
      DO 100  I=ILEV1,ILEV2
      J=J+NVREAL
      T(J)=0.D0
      IF (JJ.GE.N) GO TO 50
      J2=J+1
      J1=J2-NVREAL
      T(J)=X2*(P(J1)-P(J2))
   50 CONTINUE
      J2=J-1
      J1=J2-NVREAL
      T(J)=T(J)-X1*(P(J1)-P(J2))
  100 CONTINUE
  200 CONTINUE
  250 CONTINUE
      JJ=N
      JB=JB-1
      DO 500  L=1,LR
      N=NCOMP(L+1)
      IF (N.LT.1) GO TO 500
      DO 400  JJJ=1,N
      JJ=JJ+1
      JB=JB+2
      JR=JB+(ILEV1-2)*NVREAL
      DUM=DZ*CG(JJ)
      X1=DG(JJ)/DUM
      X2=EG(JJ)/DUM
      DO 300  I=ILEV1,ILEV2
      JR=JR+NVREAL
      JI=JR+1
      T(JR)=0.D0
      T(JI)=0.D0
      IF (JJJ.GE.N) GO TO 260
      J2=JR+2
      J1=J2-NVREAL
```

```
      T(JR)=X2*(P(J1)-P(J2))
      J2=J2+1
      J1=J1+1
      T(JI)=X2*(P(J1)-P(J2))
  260 CONTINUE
      IF (JJJ.EQ.1) GO TO 280
      J2=JR-2
      J1=J2-NVREAL
      T(JR)=T(JR)-X1*(P(J1)-P(J2))
      J2=J2+1
      J1=J1+1
      T(JI)=T(JI)-X1*(P(J1)-P(J2))
  280 CONTINUE
  300 CONTINUE
  400 CONTINUE
  500 CONTINUE
      IF (NW.EQ.0) RETURN
      WRITE (6,1000) ILEV1,ILEV2
 1000 FORMAT (1H0,10X,'I , T(',I2,') , T(',I2,') ='/)
      J1=NVREAL*(ILEV1-1)
      J3=NVREAL*(ILEV2-1)
      DO 600  I=1,NVZON
      J1=J1+1
      J3=J3+1
      WRITE (6,1010) I,T(J1),T(J3)
 1010 FORMAT (1X,I10,D20.10,20X,D20.10)
  600 CONTINUE
      NZ1=NVZON+1
      DO 700  I=NZ1,NVREAL,2
      J1=J1+1
      J2=J1+1
      J3=J3+1
      J4=J3+1
      WRITE (6,1020) I,T(J1),T(J2),T(J3),T(J4)
 1020 FORMAT (1X,I10,4D20.10)
      J1=J1+1
      J3=J3+1
  700 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VTHWIND
         .
         .
         .
      DO 25 I=ILEV1,ILEV2
      T(1,I)=TBAR(I)
   25 CONTINUE
      DO 501 J=2,NVREAL
      DO 501 I=ILEV1,ILEV2
      T(J,I)=X2(J)*(P(J+1,I-1)-P(J+1,I))-X1(J)*(P(J-1,I-1)-P(J-1,I))
  501 CONTINUE

      ENTRY THWINDO
      ILEV1=2
      ILEV2=NVERT-1
      N=NCOMP(1)
      DO 200 JJ=2,N
      DUM=DZ*CG(JJ)
      DUM=1./DUM
      X1(JJ)=DG(JJ)*DUM
      X2(JJ)=EG(JJ)*DUM
  200 CONTINUE
      JJ=N
      X2(N)=0.
      JR=N-1
      DO 500 L=1,LR
      DO 400 JJJ=1,NCOMP(L+1)
      JJ=JJ+1
      JR=JR+2
      JI=JR+1
      DUM=DZ*CG(JJ)
      DUM=1./DUM
      X1(JR)=X1(JI)=DG(JJ)*DUM
      X2(JR)=X2(JI)=EG(JJ)*DUM
  400 CONTINUE
      X2(JR)=X2(JI)=0.
  500 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE DBHEAT
      DOUBLE PRECISION PSI,ZETA,Z1,T,Z2,Q
      DOUBLE PRECISION VDERIV,TDERIV,W
      DOUBLE PRECISION TOPO,QSV
      COMMON PSI(2366),ZETA(2366),Z1(2366),T(2366),Z2(2366)
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT,YRLAG,TIME
      COMMON /CGBLK/ KD(43),CG(43),NCOMP(12),LWAVE(12),NV(43),LV(43)
      COMMON /HEATBK/ H(26),TSZON(104),TSEDDY(16),I1TSZ,I2TSZ,I1TSW,
     1    I2TSW,MERGE1,MERGE2,ZWT1(5),ZWT2(5),Q3WR(5)
      COMMON /DERIV/ VDERIV(2366),TDERIV(2366),W(2366)
      COMMON /WORKBK/ Q(2366)
      COMMON /OROGRA/ TOPO(91),QSV(105),HTSVE(2366)
      COMMON /TEMPBK/ ADVSV(182)
      DIMENSION X(7)
      DATA NW /0/
      DATA IPHASE /0/
C
C---------------------- METHOD I ------------------------------------
C
C     1.  HEAT DUE TO O3 ABSORPTION OF SW.
C
      MGM2=MERGE2
      IF (MERGE2.EQ.0) MGM2=I1TSZ-1
      CALL O3HEAT (ILEV1,MGM2)
      IF (MGM2.LT.MERGE2) MERGE2=MGM2
C
C     2.  ADD HEATING DUE TO NEWTONIAN COOLING (ABOVE Z = 4.5).
C
      MGM1=MERGE1-1
      IF (MERGE1.EQ.0) MGM1=I1TSZ-1
      DO 100  ILEV=ILEV1,MGM1
      K=(ILEV-1)*NVREAL+1
      Y=H(ILEV)
      DO 75  I=2,NVREAL
      K=K+1
      Q(K)=Q(K)-Y*T(K)
   75 CONTINUE
  100 CONTINUE
C
C---------------------- METHOD II -----------------------------------
C
C     COMPUTE EMPIRICALLY DERIVED TROPOSPHERIC AND LOWER STRATOSPHERIC
C     HEATING (BELOW Z = 4.5).
C
      IF (IPHASE.GT.0) GO TO 200
C
C     1.  COMPUTE SEASONAL PHASE VELOCITY.
C
      ANGVEL=1./720.
      PI=4.*ATAN(1.)
C
C     TROPOSPHERIC HEATING TIME DIFFERENCE (IN DAYS).
C
      TROPLG=-30.
C
C     TROPLG (NON-DIMENSIONAL).
C
      TROPLG=4.*PI*TROPLG
      IPHASE=100
  200 CONTINUE
```

```
      PHASE=ANGVEL*(TIME+YRLAG+TROPLG)
      PHASE=SIN(PHASE)
      TIME=TIME+DT
C
C     2.  DETERMINE HEATING IN ZONAL COMPONENTS.
C
      IL=MERGE2+1
      IF (MERGE2.EQ.0) IL=I1TSZ
      IH=I2TSZ
      DO 230  I=1,7,2
  230 X(I)=1.
      DO 240  I=2,7,2
  240 X(I)=PHASE
      N=NCOMP(1)
      NH=5
      IF (NH.GT.N) NH=N
      DO 300  ILEV=IL,IH
      Y=H(ILEV)
      I=(ILEV-1)*NVREAL
      DO 260  J=2,NH
      JJ=(J-2)*26+ILEV
  260 Q(I+J)=Y*(X(J)*TSZON(JJ)-T(I+J))
      IF (NH.G .N) GO TO 280
      NH1=NH+1
      DO 270  J=NH1,N
  270 Q(I+J)=-Y*T(I+J)
  280 CONTINUE
  300 CONTINUE
C
C     3.  DETERMINE NONZONAL HEATING.
C
      IL=MERGE2+1
      IH=I1TSW-1
      IF (I1TSW.LT.MERGE2) IH=NVERT-1
      DO 330  ILEV=IL,IH
      NZ1=NVZON+1
      Y=H(ILEV
      KK=(ILEV-1)*NVREAL+NVZON
      DO 330  J=NZ1,NVREAL
      KK=KK+1
  330 Q(KK)=-Y*T(KK)
      IL=I1TSW
      IH=I2TSW
      IF (I1TSW.EQ.0) GO TO 500
      DO 450  ILEV=IL,IH
      Y=H(ILEV)
      KK=(ILEV-1)*NVREAL+NCOMP(1)
      IJ=0
      DO 400  LL=1,LR
      N=NCOMP(LL+1)
      IF (N.EQ.0) GO TO 400
      NH=0
      IF (LL.GT.2) GO TO 360
      NH=4
      IF (NH.GT.N) NH=N
```

```fortran
      DO 350  J=1,NH
      KK=KK+1
      IJ=IJ+1
      Q(KK)=Y*(X(J)*TSEDDY(IJ)-T(KK))
      KK=KK+1
      IJ=IJ+1
  350 Q(KK)=Y*(X(J)*TSEDDY(IJ)-T(KK))
      IF (NH.GE.N) GO TO 400
  360 NH1=NH+1
      DO 370  J=NH1,N
      KK=KK+1
      Q(KK)=-Y*T(KK)
      KK=KK+1
  370 Q(KK)=-Y*T(KK)
  400 CONTINUE
  450 CONTINUE
  500 CONTINUE
C
C--------------------- OVERLAP ---------------------------------
C
C     DETERMINE HEATING IN OVERLAP AREA BY LINEARLY WEIGHTED
C     COMBINATION OF METHOD I AND METHOD II.
C
      IF (MERGE1.EQ.0) GO TO 610
      N=NCOMP(1)
      NH=5
      IF (NH.GT.N) NH=N
      IJ=0
      DO 600  ILEV=MERGE1,MERGE2
      IJ=IJ+1
      K=(ILEV-1)*NVREAL
      Y=H(ILEV)
      DO 530  J=2,NH
      JJ=(J-2)*26+ILEV
      Q(K+J)=ZWT1(IJ)*(Q(K+J)-Y*T(K+J))+ZWT2(IJ)*Y*(X(J)*TSZON(JJ)
     1     -T(K+J))
  530 CONTINUE
      IF (NH.GE.N) GO TO 550
      NH1=NH+1
      DO 540  J=NH1,N
      Q(K+J)=ZWT1(IJ)*(Q(K+J)-Y*T(K+J))-ZWT2(IJ)*Y*T(K+J)
  540 CONTINUE
  550 CONTINUE
      IF (N.LT.3) GO TO 555
      Q(K+3)=Q(K+3)+ZWT1(IJ)*Q3WR(IJ)
  555 CONTINUE
      K=K+N
      DO 590  LL=1,LR
      NN=NCOMP LL+1)
      IF (NN.EQ.0) GO TO 590
      DO 560  J=1,NN
      K=K+1
      Q(K)=ZWT1(IJ)*(Q(K)-Y*T(K))-ZWT2(IJ)*Y*T(K)
      K=K+1
      Q(K)=ZWT1(IJ)*(Q(K)-Y*T(K))-ZWT2(IJ)*Y*T(K)
  560 CONTINUE
  590 CONTINUE
  600 CONTINUE
  610 CONTINUE
```

```
      IL=(ILEV1-1)*NVREAL+1
      IH=ILEV2*NVREAL
      DO 630  I=IL,IH
  630 TDERIV(I)=TDERIV(I)+Q(I)
C
C     SAVE HEATING COEFFICIENTS IN HTSVE FOR OUTPUT.
C
      DO 640  I=IL,IH
  640 HTSVE(I)=Q(I)
      IF (NW.EQ.0) RETURN
      IL1=ILEV1
      IL2=MERGE1
      IL3=MERGE2
      IL4=ILEV2
      IF (IL2.GT.IL1) GO TO 650
      IL2=(IL1+IL4)/2
      IL3=IL2+1
  650 CONTINUE
      WRITE (6,1000) IL1,IL2,IL3,IL4
 1000 FORMAT (1H0,10X,'DIABATIC HEATING AT LEVELS ',4I5)
      K1=(IL1-1)*NVREAL
      K2=(IL2-1)*NVREAL
      K3=(IL3-1)*NVREAL
      K4=(IL4-1)*NVREAL
      N=NCOMP(1)
      DO 700  I=1,N
      K1=K1+1
      K2=K2+1
      K3=K3+1
      K4=K4+1
      WRITE (6,1010) I,Q(K1),Q(K2),Q(K3),Q(K4)
 1010 FORMAT (2X,I5,E15.7,15X,E15.7,15X,E15.7,15X,E15.7)
  700 CONTINUE
      K=N
      DO 800  LL=1,LR
      N=NCOMP(LL+1)
      IF (N.EQ.0) GO TO 800
      DO 750  I=1,N
      K=K+1
      K1=K1+2
      K2=K2+2
      K3=K3+2
      K4=K4+2
      K1R=K1-1
      K2R=K2-1
      K3R=K3-1
      K4R=K4-1
      WRITE (6,1020) K,Q(K1R),Q(K1),Q(K2R),Q(K2),Q(K3R),Q(K3),Q(K4R),
     1    Q(K4)
 1020 FORMAT (2X,I5,8E15.7)
  750 CONTINUE
  800 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VDBHEAT
        .
        .
        .
      CALL O3HEAT

      DO 100 ILEV=ILEV1,MGM1
      DO 75 I=2,NVREAL
      Q(I,ILEV)=Q(I,ILEV)-H(ILEV)*T(I,ILEV)
 75   CONTINUE
100   CONTINUE
      PHASE=SIN(ANGVEL*(TIME+YRLAG+TROPLG))
      TIME=TIME+DT
      DO 230 I=1,7,2
      X(I)=1.
230   CONTINUE
      DO 240 I=2,7,2
      X(I)=PHASE
240   CONTINUE
      DO 300 ILEV=MERGE2+1,I2TSZ
      DO 260 J=2,5
      Q(J,ILEV)=H(ILEV)*(X(J)*TSZON(ILEV,J-2)-T(J,ILEV))
260   CONTINUE
      DO 270 J=6,NCOMP(1)
      Q(J,ILEV)=-H(ILEV)*T(J,ILEV)
270   CONTINUE
300   CONTINUE
      DO 330 ILEV=MERGE2+1,I1TSW-1
      DO 330 J=NZ1,NVREAL
      Q(J,ILEV)=-H(ILEV)*T(J,ILEV)
330   CONTINUE
      DO 450 ILEV=I1TSW,I2TSW
      IJ=1
      DO 400 LL=1,2
      DO 350 J=1,4
      JJ=2*J-NZ1
      Q(JJ,ILEV)=H(ILEV)*(X(J)*TSEDDY(IJ )-T(JJ,ILEV))
      Q(JJ+1,ILEV)=H(ILEV)*(X(J)*TSEDDY(IJ+1 )-T(JJ+1,ILEV))
      IJ=IJ+2
350   CONTINUE
      DO 370 J=5,NCOMP(LL+1)
      JJ=2*J-NZ1
      Q(JJ,ILEV)=-H(ILEV)*T(JJ,ILEV)
      Q(JJ+1,ILEV)=-H(ILEV)*T(JJ+1,ILEV)
370   CONTINUE
400   CONTINUE
      DO 401 LL=3,LR
      DO 401 J=1,NCOMP(LL+1)
      JJ=2*J-NZ1
      Q(JJ,ILEV)=-H(ILEV)*T(JJ,ILEV)
      Q(JJ+1,ILEV)=-H(ILEV)*T(JJ+1,ILEV)
401   CONTINUE
450   CONTINUE
```

```
      IJ=0
      DO 600 ILEV=MERGE1,MERGE2
      IJ=IJ+1
      DO 530 J=2,5
      Q(J,ILEV)=ZWT1(IJ)*(Q(J,ILEV)-H(ILEV)*T(J,ILEV))
     1          +ZWT2(IJ)*H(ILEV)*(X(J)*TSZON(ILEV,J-2)-T(J,ILEV))
  530 CONTINUE
      DO 540 J=6,NCOMP(1)
      Q(J,ILEV)=ZWT1(IJ)*(Q(J,ILEV)-H(ILEV)*T(J,ILEV))
     1          -ZWT2(IJ)*H(ILEV)*T(J,ILEV)
  540 CONTINUE
      Q(3,ILEV)=Q(3,ILEV)+ZWT1(IJ)*Q3WR(IJ)
      DO 590 LL=1,LR
      DO 560 J=1,NCOMP(LL+1)
      JJ=2*J-NZ1
      Q(JJ,ILEV)=ZWT1(IJ)*(Q(JJ,ILEV))-H(ILEV)*T(JJ,ILEV))
     1          -ZWT2(IJ)*H(ILEV)*T(JJ,ILEV)
      Q(JJ+1,ILEV)=ZWT1(IJ)*(Q(JJ+1,ILEV)-H(ILEV)*T(JJ+1,ILEV))
     1          -ZWT2(IJ)*H(ILEV)*T(JJ+1,ILEV)
  560 CONTINUE
  590 CONTINUE
  600 CONTINUE
      DO 630 I=IL,IH
      TDERIV(I)=TDERIV(I)+Q(I)
      HTSVE(I)=Q(I)
  630 CONTINUE
      RETURN

      ENTRY DBHEAT0
      NZ1=NVZON+1
      MGM1=MERGE1-1
      ANGVEL=1./720.
      PI=4.*ATAN(1.)
      TROPLG=-30.*4.*PI
      IL=(ILEV1-1)*NVREAL+1
      ILEV2=NVREAL
      RETURN
      END
```

```
      SUBROUTINE O3HEAT (I1,I2)
      DOUBLE PRECISION P,Z,Z1,T,Z2,X3,Z3
      DOUBLE PRECISION XOX
      DOUBLE PRECISION TOPO,QSV,QT,SPACE,DXO3DT
      COMMON P(2366),Z(2366),Z1(2366),T(2366),Z2(2366),X3(2366),Z3(2366)
      COMMON / ONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT,YRLAG,TIME
      COMMON /OROGRA/ TOPO(91),QSV(105)
      COMMON /SPECIE/ X3GRD(6240),CNO3(6240),XNO2(330)
      COMMON/QJBLK/NZJ,L103,COLO3(26),LEVPCM,LEVDYN
      COMMON /WORKBK/ QT(2366),SPACE(5554)
      COMMON /CHEM/ XNEUT(26),TEMP(6240)
      COMMON /FTCST/ NLON,NLAT,NGRID
      COMMON /BARBLK/ TBAR(26),SIGMA(26),DNDTBR(26),DIFFM(26),DIFFX(26)
      COMMON /GENER/ DXO3DT(2366)
       DIMENSION DATAIM(6240),DX3GRD(1),QTGRD(1)
      DIMENSION XOX(1)
      EQUIVALENCE(TEMP(1),XOX(1))
      EQUIVALENCE(QT(1),DX3GRD(1)),(SPACE(2915),QTGRD(1))
      IL1SV=ILEV1
      IL2SV=ILEV2
C
C     T , X3 SPECTRAL FIELDS FOR LEVELS L103 THRU NZJ TO GRID FIELDS
C     TEMP , X3GRD.
C
      ILEV1=L103
      ILEV2=NZJ
      NLEV=NZJ-L103+1
      ILSPC=(L103-1)*NVREAL+1
      ILGRD=(L103-1)*NGRID+1
      CALL SPCGD1 (T(ILSPC),TEMP(ILGRD),DATAIM,NLEV)
      CALL SPCGD1 (X3(ILSPC),X3GRD(ILGRD),DATAIM,NLEV)
      IHGRD=ILGRD-1+NLEV*NGRID
      DO 50 I=ILGRD,IHGRD
      IF(X3GRD(I).LT.0.) X3GRD(I)=0.
   50 CONTINUE
```

```
      CALL DX3CHM
C
C     DX3GRD , QTGRD GRID FIELDS FOR LEVELS L103 THRU NZJ (DX3GRD) AND
C     LEVELS L103 THRU I2 (QTGRD) TO SPECTRAL FIELDS DX03DT , QT.
C
      IF (I2.GT.NZJ) I2=NZJ
      ILEV2=NZJ
      NLEV=NZJ-L103+1
      CALL GDSPC1(DX03DT(ILSPC),DX3GRD(ILGRD),DATAIM,NLEV)
      NLEV=1+LEVPCM-L103
      CALL GDSPC1(XOX(ILSPC),X3GRD(ILGRD),DATAIM,NLEV)
      ILEV2=I2
      NLEV=I2-L103+1
      CALL GDSPC1 (QT(ILSPC),QTGRD(ILGRD),DATAIM,NLEV)
C
C     ADD CONSTANT ZONAL HEATING FOR TOP LEVELS.
C
      ILEV1=IL1SV
      ILEV2=IL2SV
      NZ1=NVZON+1
      K=(ILEV1-1)*NVREAL
      KK=(ILEV1-1)*NVZON
      LEV2=L103-1
      IF (LEV2.LT.ILEV1) GO TO 300
      ANGVEL=1./720.
      PHASE=ANGVEL*(TIME+YRLAG)
      PHASE=SIN(PHASE)
      DO 200  LEV=ILEV1,LEV2
      K=(LEV-1)*NVREAL
      KK=(LEV-1)*NVZON
      DO 100  J=1,NVZON,2
      K2=KK+J
      K1=K+J
  100 QT(K1)=QSV(K2)
      DO 125  J=2,NVZON,2
      K2=KK+J
      K1=K+J
  125 QT(K1)=PHASE*QSV(K2)
      DO 150  J=NZ1,NVREAL
      K1=K+J
  150 QT(K1)=0.D0
  200 CONTINUE
  300 CONTINUE
C
C     PLACE MEAN (1/XN)*(DN/DT) VALUES AT EACH LEVEL IN DNDTBR(LEV)
C     FOR OUTPUT.
C
      DO 400  J=L103,NZJ
      K=(J-1)*NVREAL+1
  400 DNDTBR(J)=DX03DT(K)
      RETURN
      END
```

```
      SUBROUTINE VO3HEAT
        .
        .
        .
      COMMON /HEATBK/ . . . ,MERGE2. . . .

      CALL SPCGD1(T(ILSPC),TEMP(ILGRD),DATAIM,NLEV)

      CALL SPCGD1(X3(ILSPC),X3GRD(ILGRD),DATAIM,NLEV)

      DO 50 I=ILGRD,IHGRD
      IF (X3GRD(I).LT.0.) X3GRD(I)=0.
   50 CONTINUE
      IL1SV=ILEV1
      IL2SV=ILEV2
      ILEV1=L103
      ILEV2=NZJ

      CALL DX3CHM

      CALL GDSPC1(DXO3DT(ILSPC),DX3GRD(ILGRD),DATAIM,NLEV)

      CALL GDSPC1(XOX(ILSPC),X3GRD(ILGRD),DATAIM,NLEVPCM)

      CALL GDSPC1(QT(ILSPC),QTGRD(ILGRD),DATAIM,NLEVMGM2)

      ILEV1=IL1SV
      ILEV2=IL2SV
      PHASE=SIN(ANGVEL*(TIME+YRLAG))
      DO 200 LEV=ILEV1,LEV2
      DO 100 J=1,NVZON,2
      QT(J,LEV)=QSV(J,LEV)
  100 CONTINUE
      DO 125 J=2,NVZON,2
      QT(J,LEV)=QSV(J,LEV)*PHASE
  125 CONTINUE
      DO 150 J=NZ1,NVREAL
      QT(J,LEV)=0.
  150 CONTINUE
  200 CONTINUE
      DO 400 J=L103,NZJ
      DNDTBR(J)=DXO3DT(1,J)
  400 CONTINUE
      RETURN

      ENTRY O3HEAT0
      NLEV=NZJ-L103+1
      ILSPC= . . .
      ILGRD= . . .
      IHGRD= . . .
      NLEVPCM=LEVPCM-L103+1
      NLEVMGM2=MERGE2-L103+1
      NZ1=NVZON+1
      LEV2=L103-1
      ANGVEL=1./720.
      IF (LEV2.LT.ILEV1) CALL SOS
      RETURN
      END
```

```
      SUBROUTINE O3COL
C
C     COMPUTATION OF OZONE COLUMN DENSITY IN
C     CM-2(1 CM COL=2.682*10**19CM-2)
C
      COMMON /CONSTS/ INDEX,NR,LR,INS,INSZ,KINT,ILEV1,ILEV2,NVERT,
     1    NRTP,LRTP,NTYPE,NVECT,NVREAL,NVZON,NCYC,DT
      COMMON/ VRTBLK/ ZVAL(26),PVAL(26),PP(26),DZ,RV
      COMMON/ SPECIE/ XO3(6240),CNO3(6240)
      COMMON/ FTCST/ NLON,NLAT,NGRID,AR(30),BR(30)
      DIMENSION CP(3)
C
C     ESTIMATE COLUMN DENSITY ABOVE HIGHEST LEVEL ASSUMING AN
C     EXPONENTIAL DECAY WITH HEIGHT-WITH A SCALE HEIGHT BASED ON THE
C     ZONAL AVERAGE OZONE CONCENTRATION.
C
      IF(ILEV1-2) 10,20,30
   10 CP(1)=12.*PP(1)/DZ
      I=0
      AXO31=0.
      AXO32=0.
      DO 300 LAT=1,NLAT
      DO 100 LON=1,NLON
      I=I+1
      AXO31=XO3(I)+AXO31
      AXO32=XO3(NGRID+I)+AXO32
  100 CONTINUE
      CHECK=RV*AXO32*0.95
      IF(CHECK.LE.AXO31) GO TO 200
      RATIO=AXO32/AXO31
      DM=ALOG(RATIO)/DZ
      CONST=CP(1)/(DM+1.0)
      I=I-NLON
      DO 150 LON=1,NLON
      I=I+1
  150 CNO3(I)=CONST*XO3(I)
      GO TO 299
  200 I=I-NLON
      DO 250 LON=1,NLON
      I=I+1
      CNO3(I)=0.0
  250 PRINT 5,I,XO3(I)
    5 FORMAT (10X,'OZONE SCALE HEIGHT IS TOO SMALL AT LEVEL',I5,5X,
     1    'XO3 = ',E10.3)
  299 CONTINUE
  300 CONTINUE
```

```
C
C     COMPUTATION OF OZONE COLUMN DENSITY ABOVE OTHER LEVELS USING
C     A QUADRATIC CURVE FIT TO THE VERTICAL PROFILE OF OZONE
      BETWEEN LEVELS.
C
   20 I=NGRID
      CP(1)=5.*PP(2)
      CP(2)=8.0*PP(1)
      DO 400  L=1,NGRID
      I=I+1
      I1=L
  400 CNO3(I)=CP(1)*XO3(I)+CP(2)*XO3(I1)+CNO3(I1)
      IL1=3
      GO TO 35
   30 I=NGRID*(ILEV1-1)
      IL1=ILEV1
   35 DO 600 J=IL1,ILEV2
      J1=J-1
      J2=J1-1
      CP(1)=5.0*PP(J)
      CP(2)=8.0*PP(J1)
      CP(3)=PP(J2)
      DO 500  L=1,NGRID
      I=I+1
      I1=I-NGRID
      I2=I1-NGRID
  500 CNO3(I)=CP(1)*XO3(I)+CP(2)*XO3(I1)-CP(3)*XO3(I2)+CNO3(I1)
  600 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE VO3COL
         .
         .
         .
      DO 400 K=1,NLAT
      DO 400 I=1,NLON
      CNO3(I,K,2)=CP1*XO3(I,K,2)+CP2*XO3(I,K,1)+CNO3(I,K,1)
  400 CONTINUE
      DO 600 J=3,ILEV2
      DO 500 K=1,NLAT
      DO 500 I=1,NLON
      CNO3(I,K,J)=CP(J,1)*XO3(I,K,J)+CP(J,2)*XO3(I,K,J-1)
     1            -CP(J,3)*XO3(I,K,J-2)+CNO3(I,K,J-1)
  500 CONTINUE
  600 CONTINUE
      RETURN

      ENTRY O3COL0
      DO 601 J=ILEV1,ILEV2
      CP(J,1)=5.*PP(J)
      CP(J,2)=8.*PP(J-1)
      CP(J,3)=PP(J-2)
  601 CONTINUE
      CP1=5.*PP(2)
      CP2=8.*PP(1)
      RETURN
      END
```

DIVISION 5

TECHNOLOGY SURVEY UPDATE

DIVISION 5

TECHNOLOGY SURVEY UPDATE

INTRODUCTION

This segment of the report provides current information as an update of the estimates and prognostications provided in reference 1, on the general subject of circuit technologies, primarily semiconductor. A number of other bodies of technology, which might be called "System Technologies", are also critical to the FMP design, and their significance should not be lost. Their rate of change, however, tends to be less dramatic and less public; for this reason the following material dwells on circuitry, emphasizing the changes perceived since the previous report.

These can be summarized as follows:

- In the mainline circuitry (ECL) the scales of integration and possibilities have become more defined. There is, of course, still considerable margin in the time dimension, but the picture is sharper than a year ago.
- Of the long-shot logic technologies, Gallium Arsenide (GaAs) is benefitting from increased investment. As such it is worthy of somewhat closer attention. Cryogenic options remain too esoteric to warrant serious concern.
- Of the potential auxiliary memory technolgies, CCDs are still the most plausible, although their availability for the target time frame has become more questionable. The availability of 64K dynamic RAMs (DRAMs) on the other hand is more certain, albeit at costs which may be higher than is acceptable.
- An "intermediate" memory has been postulated for the proposed FMP. It will utilize only established low-cost technologies.

Some elaboration on these points follows.

CRITICAL CIRCUIT TECHNOLOGIES

The previous studies (refs. 1, 2) developed the decision that the FMP schedule is best served by use of high-speed ECL logic such as is now (1979) coming into production. The Fairchild version is called F200K, and the internal CDC designation is LSI-168.

As this technology matures natural improvements in cost, reliability and speed will evolve. The next most likely major change would be to a considerably larger scale of integration with some modest initial speed improvement. A change such as this, however, requires a major overhaul of the CAD/CAM (computer aided design/computer aided manufacture) support system plus a step up in semiconductor technology. To the extent such techniques and circuitry are available for use, they may be invoked in critical areas. However, serious consideration for their use cannot be prudently planned in the time scales proposed for the FMP development. Table 1 collects the LSI products in the high speed (ECL) technology, and projects some estimates of availability of the next most significant steps. The final entry (availability 1985) is quite conjectural at this stage. Semiconductor technologists tend to think in terms of three-year increments, so data gathered from them often shows this sort of expected cycles. In point of fact the technology does not move in sudden jumps, but inches along on a broad front. Enough of these breakthroughs are known to be pending, however, that the progress has reasonable predictability. The time scales have a way of stretching out, however, particularly as reasonably large-scale manufacturability is required.

TABLE 1
ECL LSI RELATED PRODUCTS

PRODUCT	YEAR* INTRODUCED	EQUIVALENT GATES/CHIP	STAGE DELAY (ps)	STATUS
Amdahl Gate Array	73	100	750	Custom
CDC MOT Gate Array	75	190	900	Discontinued
CDC/F200K Gate Array	77	250	650	In proto production
Siemens F100K Gate Array	77	500	750	Final development
Motorola 10K Macro Cell Array	(79)	750	1200	In development
FSC 8-Bit Slice Set	(79)	2K-8K	650	Custom set of four types (not an array)
Next Generation Gate Array	(82)	(1.5K-2K)	(500)	In exploration stages at FSC, MOT, NATL.
Follow-On Gate Array	(85)	(5K-7K)	(250)	Prediction by suppliers

*Numbers in parenthesis are anticipated.

Suppliers such as Motorola, Fairchild, and National presently see 1000-2000 gate equivalents being practical in preproduction quantities by 1982. Although processing differences exist, all project use of some form of oxide isolated, walled emitter ECL process on a die size of less than 8mm, the current practical limit for projection step and repeat photolithography. All agree that additional advances in photolithography to 1 micron or less and improved metalization processes are required to achieve the 1985 objectives of some 6000 equivalent gates per die.

Other forms of logic have demonstrated subnanosecond performance and must be given consideration. One form, Josephson switches, has demonstrated sub-100 ps delays but require a super-cooled (4-5 degrees Kelvin) ambient environment. CDC, to date, has monitored the progress of this technology only via periodical reviews. The feasibility of conducting actual Josephson switch experiments is currently under investigation. A second form of subnanosecond delay technology that has been demonstrated in R&D facilities at Rockwell, Hewlett-Packard, RCA, TRW, Motorola, and others utilizes MOS technology and Gallium Arsenide (GaAs) material. Because of the superior mobility of GaAs, 5-6 times that of silicon, delays in ring counter form of 75-150 ps have been demonstrated. The device operates with low power (microamperes current) and in normal ambient temperatures. GaAs devices, in fact, have superior quality to silicon at high temperatures. Material uniformity difficulties, as well as the difficulty in growing necessary oxide coatings over the wafer surface, have hampered progress in this technology's growth. Recent advances in ion implantation have helped, and densities up to 100 gates/die are considered achievable in CY79. Because of the similarities to silicon, the projected superior performance while operating at lower power, the MOS-like circuit packaging densities, and the recent interest shown by major suppliers, efforts to watch this technology more seriously are warranted.

Recent projections for NMOS, CMOS and I^2L technologies might suggest that these logic families are to be considered legitimate high performance candidates in the near future. However, they tend to "bottom out" speed-wise in the 1-3 ns range. For applications which demand the ultimate speed, they cannot be seriously considered unless some major breakthroughs occur in architecture which can overcome this handicap. Because the low speed-power product of these technologies enables a remarkable scale of integration, and because of their value in memory products, it is felt proper to spend some time on them here.

The present candidates include: one bipolar prospect (I^2L), improved NMOS technology in two forms titled HMOS and VMOS, and the CMOS-SOS, or CMOS-SOIS, technologies. I^2L is in a form of inverted transistor technology utilized by several suppliers including Signetics in a general gate array of some 400 gates, and FSC in the 9440 16-bit microprocessor family (over 7-8K gates/chip in production). CDC also manufactures some arrays of this type internally. HMOS is a form of scaled geometry and scaled processing NMOS technology aimed at high performance and high density applications. Intel utilizes this process in their sub-50 ns static RAM (SRAM) products and their 8086 16-bit microprocessor. VMOS is an alternative to HMOS utilizing the attributes of etching pits into silicon to form fine gate geometries by implantations/diffusions rather than surface geometries as done in HMOS. AMI utilizes this process in SRAM products for high density as well as performance. CMOS refers to complementary MOS. This implies that the circuitry always has one "off" device in the ground-to-power-bus chain. The switches (ignoring minor leakage currents) have power dissipated only during logic switching periods. The device gate-to-gate on-chip capacitance is reduced further by utilizing either sapphire as the insulating substrate (silicon on sapphire) or oxide isolation between switches (silicon on insulated substrate).

I^2L and HMOS/VMOS also utilize oxide insulated device separation for reduced capacitance.

Conservative projections show 10K-50K gate equivalents becoming reasonable by the early 80's with moderate improvement in photolithography (1 micron spacings and widths). In addition, stage delays are projected to approach, if not improve below, 1 ns. Presently, these technologies in R&D design applications (not perfectly designed ring counters for optimum performance) offer impressive 2-4 ns stage delays.

Figures 1 and 2 illustrate the expected trends in two "best-bet" technologies (ECL and CMOS/I^2L) plus the possible evolution of what CDC considers to be the best wild-card technology, GaAs.

AUXILIARY MEMORY TECHNOLOGIES

To review, the term auxiliary memory technologies has been used to denote the various storage technologies which promise significant cost improvements relative to RAMs, yet with considerably improved access time vis-a-vis rotating magnetic storage. The candidates have been considered to be charge coupled devices (CCD), magnetic bubbles (MBM), and electron beam storage (EBAM).

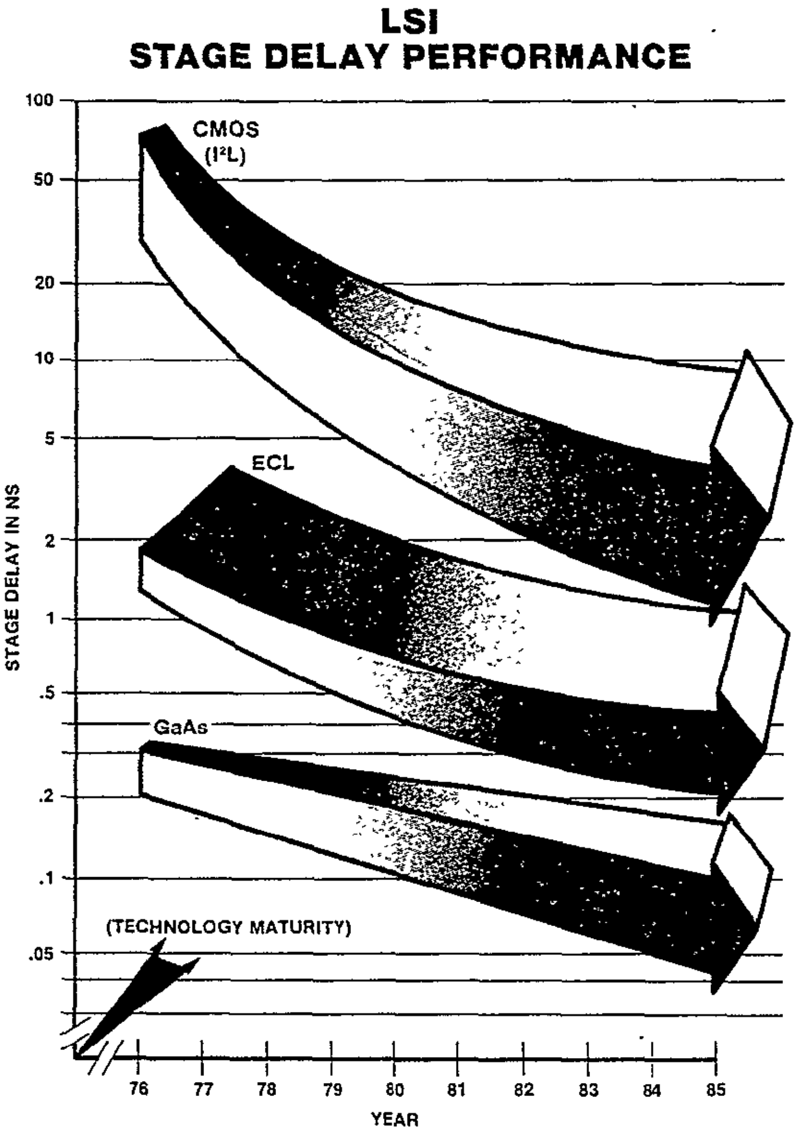


Figure 1. LSI Stage Delay Performance

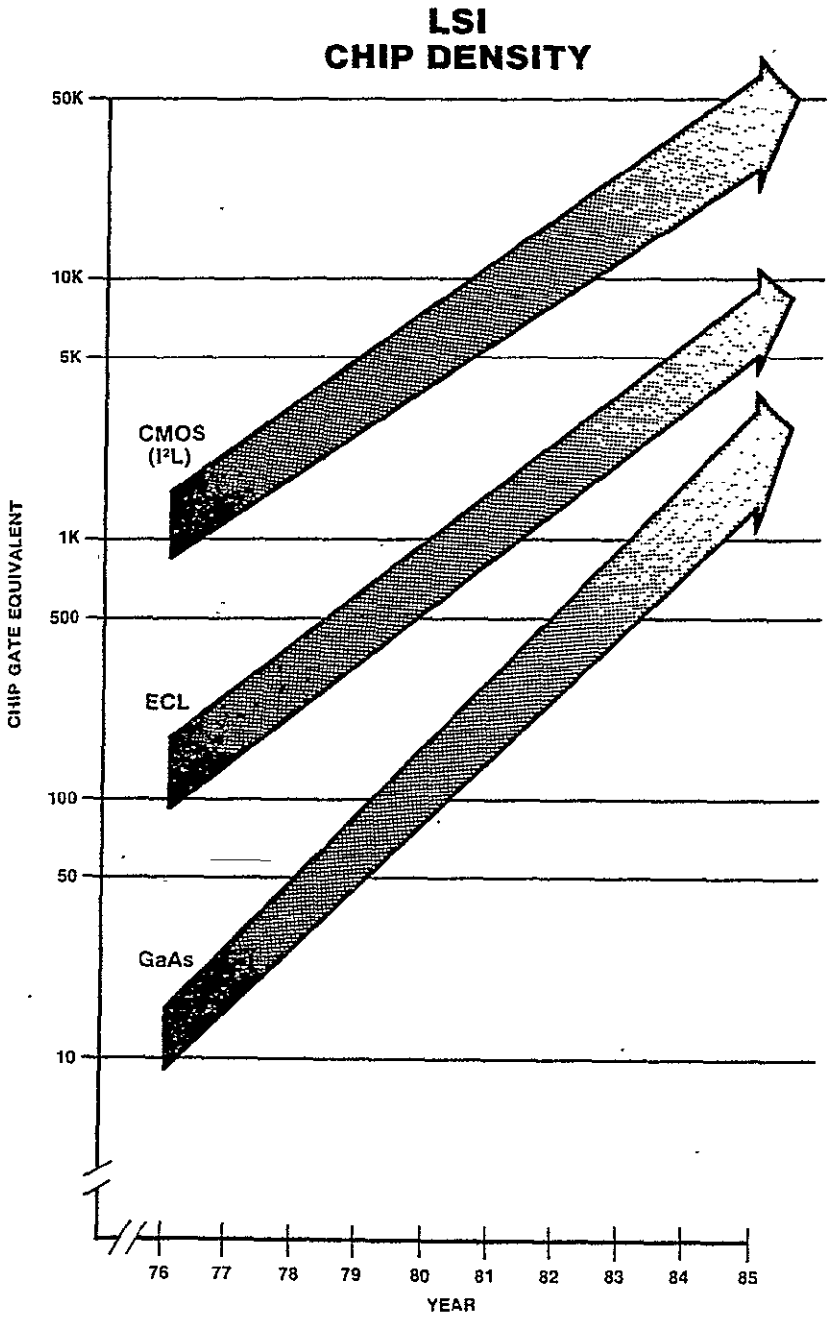


Figure 2. LSI Chip Density

None of these has kept pace with its protagonists' hopes or projections over the past five years. Commercial products have appeared for both CCDs and MBMs. In both cases the products, while quite capable, have proved to be considerably more limited in application than hoped for. Nevertheless they have found some valuable use, and as such must be judged to have been "born". Table 2 summarizes the known extant examples, as well as some current predictions.

It is quite clear that auxiliary memory technologies must achieve enough volume to drive their cost below the DRAM competition, a classic chicken and egg situation. If this is accomplished for CCDs they clearly are the best bet for the FMP Backing Store, a memory which is a significant requirement in order for the FMP to achieve its expected system performance. The probability of suitable CCDs being available to build the Backing Store is quite acceptable, although not completely assured; at this point the cost may prove to be a larger hurdle.

Magnetic bubble technology is driving increasingly toward lower cost, at the expense of speed. This comes about because of the need to take advantage of MBMs special properties, e.g., non-volatility, as well as the particular markets these open up. Several semiconductor houses have made significant investments in MBM recently, indicating a growing appreciation for the potential markets. This type of investment is needed to drive the costs down. So in the case of bubbles, the availability and part costs look reasonably promising for the FMP Backing Store; but the speed expectations pretty well rule them out.

EBAM's continue to attract some research investment as a long shot possibility. The DARPA-sponsored interest in this area has, however, been abandoned as of this fiscal year.

TABLE 2

AUXILIARY MEMORY TECHNOLOGY

	Charge Coupled Devices			Magnetic Bubble Devices		
CAPACITY	65K	256K	1M	256K	1M	4M
Availability						
Test chips	-	2Q79	1981-82	4Q77	1Q78	1981
Prototypes	2Q77	1Q80	1983-84	4Q78	1Q80	Late 82
Production	1978	1981-82	--	1981	1982	1984
Volume cost advantage*	Mid 80	1984-85	--	1980	1982	1984
Cost ($/M byte)**	800	120	--	800	250	40
Access time (avg)	400 us	400 us	400 us	2.5-5 ms	5-10 ms	5-10 ms
Data Rate (Mbits/sec)	5	5	5	.1-.2	.1-.2	.1-.2
Power (mw)/package	300	200	200	500-1000	500-1000	500-1000
Material	<-------Silicon------->			<-------Garnet------->		
Die Size (mil^2)	37K	35-40K	35-40K	50-60K	50-60K	50-60K
Packaging	<-16 pin DIP-- --0.350 x 0.450 CCC->			<-----Non-STD DIP----->		
Geometries						
Minimum feature	6u	3u	1-2u	1.0-1.5u	1.0u	1.0u
Oxide thickness	1000 A	600-700A	400-600A	--	--	--
Nominal layer thickness	--	--	--	5000A	5000A	5000A
Volatile	yes	yes	yes	no	no	no
Start/Stop capability	no	no	no	yes	yes	yes

---

*Volume cost advantage occurs after the steep portion of the learning has been completed.

**Cost/price estimates are highly speculative until technology market are established.

DIVISION 6

NASF RELIABILITY-AVAILABILITY EVALUATION

DIVISION 6

NASF RELIABILITY-AVAILABILITY EVALUATION

This evaluation incorporates observed MTBF for standard Control Data computing equipment based upon most recently filed data, and estimates of inherent MTBF for the FMP based upon the most recent reliability evaluation of the CYBER 203 (STAR-100A). Several assumptions have been made in the FMP evaluation and the total system evaluation to provide meaningful structure to the reliability models and to simplify the computations. Assumptions always introduce error; however, where assumptions and/or estimates were made, the effort was to bias the decisions toward the worst-case condition which should yield conservative MTBF and availability estimates.

FMP Reliability Evaluation

The failure rate and MTBF estimates of the FMP and its units are shown in table 1. Two types of failure rates, elemental and functional, are derived since the memory units and a large part of the data transfer paths utilize single-error correction/double-error detection (SECDED) logic. The definitions of these two types of failure rates are given in CDC-STD 1.12.999 Glossary of Reliability, Availablity and Maintainability Terms (included as appendix A).

The FMP logic units are assumed to be similar in structure and logic design to the CYBER 203 and therefore the reliability model uses the same methods as that of the CYBER 203. The CYBER 203 reliability prediction is based upon the detailed final equipment configuration. The reliability model is composed of reliability modules of the basic configuration building blocks. These modules are derived from the CYBER 203 structure in a way which combines the physical entities into a functional module. For example, the LSI device reliability module is composed of the LSI device and its associated capacitors, solder joints, connector and proportionate number of terminators. In a like manner, reliability modules are defined for the LSI boards, F100K circuits, and so on. The proportionate number of parts, such as terminators, vias, coax connectors, and the like, are allocated on the basis of the average distribution in the CYBER 203. Appendix B defines the reliability modules. The component part failure rates for some of the major parts (e.g. the LSI device, the storage device, etc.) have changed since the last report and their new failure rates are given in appendix C. Failure rates for other parts are taken from CDC-STD 1.12.020 Component/Piece Part Failure Rates (included as appendix D).

Appendix E is the derivation of the elemental failure rates of the FMP units utilizing engineering design estimates and the reliablity modules of appendix B. The functional failure rates of the logic units are derived by the method described in appendix F using the SECDED model described in appendix G.

Table 1
FMP Reliability Summary

Unit		Failure Rate* Elemental	Failure Rate* Functional
Scalar		665.5	665.5
Swap		70.8	16.7
Intermediate Map		83.6	56.6
Main Map	118.4 \		
Memory Interchange	424.6 >	697.3	425.3
Vector Streaming	154.3 /		
Vector		1,848.0	1,848.0
Stream Control		61.0	61.0
Input/Output		167.1	167.1
Main Memory		14,298.5	1,924.3
Intermediate Memory		41,666.7	1,518.8
Backing Store (262K device)		36,624.6	1,845.6
Refrigeration		409.0	409.0
Power		438.2	438.2
Total FMP failure rate*		97,030.3	9,376.1
FMP MTBF		10.3 hrs	106.7 hrs.

FMP Reliability with 65K CCD

Total with 262K CCD		97,030.3	9,373.8
Less Backing Store (262K)		36,624.6	1,845.6
60,405.7	7,528.2		
Plus Backing Store (65K)		159,185.4	8,576.0
Total FMP failure rate*		219,591.1	16,106.5
FMP MTBF		4.6 hrs.	62.1 hrs.

*Per 10^{6} device hours (10^{-6} failures per device hour).

The Main Memory is based on the design of a memory now under development utilizing a 1 x 4096 ECL memory device. Its elemental and functional failure rates and MTBFs are derived in appendix F as are those of the CCD Backing Store. The failure rates for Intermediate Memory (shown in table 1) were provided by the Control Data Division which is currently responsible for memories of that type, on which its design is based.

NASF System Reliability Evaluation

The NASF system, composed of the FMP and standard system components configured in a redundant manner, is summarized in table 2. Because of the system complexity and redundancy, and because utilization of the system does not always require every computational and data handling function to be available simultaneously, a meaningful overall reliability figure of merit (from a user viewpoint) cannot be stated. However, the availability and reliability, as may be seen by a user, can be derived if the required resources and period of use are known. An example is used to provide an estimate of what might be expected. The detail assumptions and derivations are presented in appendix H; a summary of the assumptions and results are presented here.

Assumptions

1) A remote user uses the following resources for a two-hour period:

   2551 Communications Controller
   CYBER 175 Computer
   PDCs for system interconnection
   FMD Disk Subsystem
   ECS Subsystem
   819 Disk Subsystem
   FMP

2) The FMD Disk Subsystem uses three of the four disk units.

3) The four 819 disk units assigned to the user, for practical purposes, have no immediate back-up.

Results

1) User availability (the probability that the resources are available on demand) is 98.89%

2) User reliability (the probability of completing the task in two hours) is 98.24%

Other performance characteristics of the system are:

1) The operating system critical components (the FMP, the ECS, and the CYBER 175s) may cause a system interruption on the average of once every three to four weeks. In these cases the operating system must be reloaded and all users will have to reinitiate their jobs or tasks. (See operating system critical MTBF derivation in appendix H.)

2) Something in the system fails approximately every six hours but because more than 50% are correctable, an operator or user may be inconvenienced every thirteen hours.

3) Assuming the example (appendix H) represents a typical use of the system, the applicable failure rate apparent to a user or class of users is 8804 failures per million hours or 113.6 hours MTBI. Stated another way, once every four or five days a given user may be required to restart or reinitiate the job or task being performed. Eighty-two percent of these failures are local with respect to the system, meaning only specific users are affected. The other eighteen percent are a result of a system failure affecting all current users.

Table 2
NASF Reliabily Summary

		Elemental		
System Component	Qty.	MTBF	Failure Rate	Functional MTTR*
CDC FMP	1	10.3	97030.3	1.5
CYBER 175	2	367	2725.0	1.7
CYBER 18	1	362	2762.0	2.4
(2551) Network Processor	2	1846	541.7	1.6
Programmable Device Controller	16	10000	100.0	1.5
7030 ECS	1	630	1587.3	1.8
677 Tape Unit	2	1022	978.5	1.6
679 Tape Unit	2	1022	978.5	2.2
7021 Tape Controller	1	3200	312.5	1.8
885 FMD Disk	4	5000	200.0	2.0
7155 FMD Controller	2	8000	125.0	3.0
7881 Cartridge Storage Unit	8	1730	578.0	1.5
7882 Mass Storage Transport	16	960	1041.7	1.0
7880 Mass Storage Controller	3	1970	507.6	4.0
819 Disk Drive	16	2800	357.1	2.2
7639 Disk Controller	4	5000	200.0	1.2
580 Train Printer	4	442	2262.0	2.2
405 Card Reader	2	1091	981.4	1.5
3447 Card Reader Controller	2	24000	41.7	4.0
415 Card Punch	1	1091	981.4	3.4
3446 Card Punch Controller	1	14400	69.4	2.9
8271 Transfer Switch	2	72000	13.9	0.5
3270 Switch Controller	1	12000	83.3	2.0
NASF Totals		6.39	156393.2	

*Because of system redundancy, the MTTR should not, except in the case of the FMP, be taken to be the time that the system is down when the associated equipment fails. Even in event of an FMP failure, if the failure is in a vector pipeline or in I/O (about 20% of the time) the system can be back up in minutes.

Appendix A

GLOSSARY OF RELIABILITY, AVAILABILITY, AND MAINTAINABILITY TERMS

CONTROL DATA CORPORATION

SYSTEM STANDARD

STD 1.12.999
REV A
DATE August 1978
PAGE 1 of 9

GLOSSARY OF RELIABILITY, AVAILABILITY AND MAINTAINABILITY TERMS

1.0 SCOPE

1.1 Purpose - This Standard delineates a list of terms and their definition as used in CDC on the subjects of Reliability, Availability and Maintainability (RAM). These definitions are intended to reduce inconsistancies and confusion in nomenclature.

1.2 Applicability - This standard applies to other standards in the CDC-STD 1.12.000 series.

1.3 Effectivity - This standard is effective immediately upon release.

1.4 Authority - The enforcement of this standard is in accordance with CDC-Policy 10:04:00. Waivers from this standard are only granted via the controlling document. See CDC-Policy 10:04:30. The interpreting authority for this standard is the General Manager of CDC Technical Standards.

2.0 APPLICABLE DOCUMENTS

2.1 Referenced Documents
CDC-Policy 10:04:00 CDC Technical Standards
CDC-Policy 10:04:30 Deviations or Waivers from CDC Technical Standards
CDC-STD 1.12.000 Reliability, Maintainability and Availability Standards

2.2 Related Documents
None

3.0 GLOSSARY

Not Applicable

4.0 REQUIREMENTS

Many of the terms are composed of two or more words. These terms appear with the noun first followed by a comma and all modifiers. Thus, "software installation aids" would appear as "aids, software installation" and would be found with the term starting with the letter "a".

action, repair -
A single maintenance procedure or step designed to completely correct a failure. Examples of repair actions are "replace module at Location B13" or "perform Procedure 54 to re-align the read head".

availability -
The probability that an item will perform its specified operation under stated conditions at any given time.

availability, basic -
The fraction of time that a system or product is not being repaired. It is also called intrinsic availability and inherent availability. Basic Availability in fractional notation is:

$$\text{Basic Availability (Ab)} = \frac{\text{Measurement Interval - Active repair time}}{\text{Measurement Interval}}$$

Approved by D. L. Bickel
D. L. Bickel, Vice President
Operations Services, Computer Group

availability, net -
The fraction of a system's designed and expected (potential) throughput achievable on demand during a given calendar time period. Potential throughput is that which can be achieved in the absence of interruptions and scheduled maintenance. Net availability is reduced by all lost time, both system down and degraded, during scheduled operating time and scheduled maintenance time.

$$\text{Net Availability (An)} = \frac{\text{Calendar time} - \text{lost time} - \text{scheduled maintenance time}}{\text{Calendar time}}$$

availability, user -
The fraction of a system's designed and expected (potential) throughput achievable on demand during scheduled operating time. Potential throughput is that which can be achieved in the absence of interruptions. User availability is similar to net availability except that only events occurring during scheduled operating time are counted. User availability is reduced by all lost time, both system down and degraded, during scheduled operating time caused by an interruption.

$$\text{User Availability (Au)} = \frac{\text{Scheduled Operating Time} - \text{Lost Time}}{\text{Scheduled Operating Time}}$$

average, running -

$$\text{Current month running average} = \frac{\text{Current month average} + \text{twice the previous month's running average}}{3}$$

where the running average of the first 3 months is defined as the average for the months being measured. This is done so as not to overemphasize the first month data. In cases where the current month's value is very large compared to the previous month's running average (e.g., no failures for the month) the following equation should be used:

$$\text{Current month running average} = \frac{3}{1/\text{current month average} + 2/\text{previous month's running average}}$$

call, service -
The response to a request for remedial maintenance that is attended to at the user location by one or more persons from the maintenance organization. A service call is normally occasioned by one or more incidents such as failures, misuse, or media caused failures. Service call-backs are included. Activities such as preventive maintenance and installation of modifications are excluded.

configuration, target -
The configuration which approaches the predicted typical field application of the product.

controlware -
A processor program for a particular processing unit integral to a product that provides the product a set of functional operating characteristics. Controlware is supplied as part of a product in accordance with applicable Control Data policies and procedures and is necessary for proper product operation. The programs are considered as programmed functions which may be analogous to hardware logic and are documented, maintained and supported in a similar manner.

deadstart -
The initial action taken to start a computer system when no software is resident or active in that system. Deadstart normally occurs after a total system shutdown or an unrecoverable system error and causes system initialization and operation initiation.

degradation, graceful -
The automatic and/or orderly removal of some of a system's capabilities due to a failure. This is accomplished in a manner which allows continued system operation, but with reduced capability.

detection, data error -
The process (whether by software and/or by hardware means) of recognizing that one or more bits are incorrectly transferred, stored, read or manipulated.

diagram, reliability block -
A pictorial arrangement of parts (functional blocks) which describes the separably identifiable functions of a product, equipment or system and their reliability relationships.

DPSR - Diagnostic Programming System Report -
A form, AA 4329, used by CDC maintenance software users to report maintenance software failures. The DPSR applies to released products. DPSRs are classified the same as PSR (for classification definition see PSR).

effectiveness, maintenance software -
Used with maintenance software as a specification of performance which indicates the degree with which maintenance software produces its desired result of detecting and isolating failures.

error -
The difference between an observed or calculated value and a true value, as in data error; something produced by mistake, as in design error.

error, system recovered -
The encountering of a failure or error which is detected and recovered from without 1) manual intervention, 2) loss of any system resources, 3) producing incorrect results, and 4) termination of any application abnormally. The encountering of a failure which does not result in a system down or degraded interruption.

factor, degradation -
The average percentage throughput capability lost with the specific class of failures or interruptions.

factor, duty -
The duty factor is the percentage of time an item is used. That is:

$$\text{Duty Factor} = \frac{\text{Actual Usage Time}}{\text{Scheduled Operating Time}}$$

failure -
A state of inability of an item to perform its intended function. The cause may be breakage or deterioration beyond specified limits or design errors. A failure may be a data error rate which has deteriorated beyond specified limits. Multiple encounters of the same fault/failure are a single failure. (See Interruption.)

failure, elemental -
A component failure requiring replacement or adjustment within a unit whether or not the unit ceases performing. For example a single bit failure in a single error correcting double error detecting (SECDED) memory.

failure, functional -
Any failure or combination of failures which causes a system, product or equipment to cease performing a specified function.

failure, maintenance software -
Each of the following are maintenance software failures:

- The inability to complete testing due to an error in the maintenance software program itself.

- The report of a hardware fault when no fault exists
- The incorrect identification of a hardware fault even though another fault is present in the hardware
- The inability to perform an auxiliary function due to a design error in the maintenance software

fault -
The cause of failure.

FCO -
Field Change Order. The directive to install changes to equipment after the normal manufacturing process in order that the equipment will perform to its written or implied specification.

firmware -
A physical electronic component in which a program resides that is incorporated in a product to provide a programmable mode of operation defining the product's functional characteristics. Firmware is not self-modifiable, and is subject to change or modification only by physical modification or replacement.

incident, failure -
An occurrence requiring remedial maintenance to correct a single failure.

installability -
A characteristic of design and environment which expresses an item's ability to be configured into a system in the manner specified by a customer on his location.

interruption -
The cessation of productive processing due to the encountering of a failure. An interruption is not ended until it is followed by 15 minutes or longer of productive processing.

interruption, system degraded -
The encountering of a failure which does not result in a system down interruption or a system recovered error. A system degraded interruption 1) results in an application program terminating abnormally or producing an incorrect result, or 2) requires some manual intervention or the downing of some system capability to allow recovery.

interruption, system down -
The encountering of a failure which results in none of the user applications being correctly processed. This can be recognized by a deadstart or manual intervention being required to return the system from a down state to a productive state.

isolation, error/failure -
The process by either software and/or hardware means of localizing an error or failure to a specified level.

K-factor -
A translation modifier which relates various reliability parameters. For example: (failures)(K-factor) = interruptions and (inherent MTBF)(K-factor) = observed MTBF. Specific K-factors require that the parameters being related be identified.

life -
The actual use time before a product will be scrapped or require a major refurbishment.

mainframe -
An organized collection of directly connected hardware products, equipments, parts and accessories consisting of a single central memory, one or more central processors, peripheral processors, channels and control consoles.

maintainability -
A characteristic of design and environment which is expressed as the probability that an item will be retained in or restored to a specified condition within a given period of time, when the maintenance is performed in accordance with prescribed procedures and resources.

maintenance -
Any activity to repair a product or correct supporting documentation in order to eliminate or prevent errors or failures in a post-release product.

maintenance, preventive -
A procedure of periodically checking and/or re-conditioning a system or unit to prevent or reduce the probability of failure or deterioration while in service.

maintenance, remedial -
Those activities where a technician is working on a unit or system on a customer installation to make the unit or system operational except those activities that are considered preventive maintenance, associated repair or check-out.

management, reconfiguration -
A procedure that manipulates the organization of system resources such that the system can continue to perform useful work when one or more system elements are unavailable to the user.

margin selection, maintenance software -
The capability within maintenance software to run with hardware margins selected either manually or under control of the maintenance software.

MLT - Mean Lost Time -
The average lost time per interruption or class of interruption over the time period being measured.

modularity, spares -
Packaging replaceable hardware sub-assemblies in a manner that minimizes the sum of per unit manufacturing cost plus field replacement costs.

modularity, system -
The organization of system elements such that logical functions with specified interfaces can be easily distinguished and, when necessary, logically and physically isolated from one another.

MTBF - Mean Time Between Failures -
The average time from the start of one fault or failure to the start of the next. The specific time base used is to be indicated in the context of the MTBF usage.

MTBF is Mean Time Before Failure in some reliability prediction equations. In this usage MTBF is the average time from the end of a failure to the beginning of the next failure. This meaning of MTBF is not used in the RAM standards. When non-operable time is only a few percent or less of the total time, then the two uses of MTBF are approximately equal.

MTBF, inherent -
An MTBF number derived from component failure rates (anticipated stress levels should be considered). No considerations are made for possible poor design or poor manufacturing or inadequate service.

MTBI - Mean Time Between Interruptions -
The average time from the start of one interruption to the start of the next. The specific time base used is to be indicated in the context of the MTBI usage.

MTBSC - Mean Time Between Service Call -
The average time between service calls initiated by customer request (i.e., system failures, service call backs, misuse, media caused failures, etc., excluding P.M. and installation of modifications); obtained by dividing total time by the number of service calls.

MTTR - Mean Time to Repair -
The overall average time it takes to diagnose a machine fault, repair it, and adequately verify the operation after repair; obtained by dividing the total unscheduled repair time by the number of repair incidents. This calculation average does not include associated repair, travel time or wait time.

product -
A hardware, software, or supply item that is saleable to a customer.

product set -
The complement of software (excluding the Operating System) supplied by the vendor for use by the customer in writing application programs, storing and manipulating data, e.g., language processors, SORT/MERGE, Data Management.

program, application -
Software written by a user or supplied by a vendor to solve a particular problem related to the users business, e.g., payroll, linear programming package.

program, maintenance software -
A software program that detects and/or isolates, that facilitates repair, that aids in adjustment, or that confirms repair of a hardware failure.

PSR - Program System Report -
A report used by CDC software users to report software failures and errors. PSRs are classified into categories of criticality. They are critical, urgent, serious, minor, and informational. (Similar to TAR for Hardware, firmware and controlware).

PSR, critical -
A PSR category where the reported failure results in frequent (1 or more per day) system downs and/or a major project stalled through software problems.

PSR, informational -
A PSR category in which errors in comments, coding techniques, or documentation are reported. Code change is not required.

PSR, minor -
A PSR category where reported failures result in inconsistencies or irregularities that require a code correction. (The category refers only to the urgency of the need for software maintenance). Items of inconvenience or of minor or local consequence should preferably be in this category.

PSR, serious -
Problems that definitely need to be fixed at once, but for some reason are below urgent or critical. For example, a PSR belongs in this category if the problem can be circumvented, if a local or temporary fix is available, or if it is an urgent problem that only occurs rarely or under unusual circumstances.

PSR, urgent -
Regular system crashes (more than 1 per week); substantial user difficulties. High probability of serious problems (such as bugs in error recoveries, etc.).

RAM -
Reliability, Availability and Maintainability.

recovery, data error -
The process of amending or repeating a data transfer, store, read or manipulation, which resulted in a data error, to produce the correct result.

release, field -
A term used in the RAM standards to indicate the point in an item's life when it has been certified.

reliability -
The characteristics of an item expressed as the probability that it will perform a required function under stated conditions for a stated period of time.

repair, associated -
The repair of a replaceable module, subassembly, or product after it has been replaced by a like module in the user's system. Example, the action of repairing spare parts after the product has been returned to service.

software, fail-soft maintenance -
Maintenance software which is an integral part of the operating system software and other system software and which provides failure management capabilities; i.e., dynamic hardware failure detection, error logging or recovery activities.

software, hardware checkout -
Software designed for and used only during the engineering or manufacturing checkout of various hardware devices. It may be required when the checkout requirements cannot be satisfied by Hardware Design Verification Software.

software, hardware design verification -
Software designed for and used in the process of hardware or microcode design verification testing. It may be required where design verification requirements cannot be met by Off-line Maintenance Software.

software, in-line maintenance -
Maintenance software designed for use in field maintenance of hardware which operates within a subsystem independent of the operating system and which may be used concurrently with customer use of the subsystem.

software, maintenance -
Any computer program code and associated documentation, used for maintenance of released products, that detects and/or isolates failures, facilitates repair, aids in adjustment, and confirms normal operation of hardware.

software, off-line maintenance -
Maintenance software designed for use in the field maintenance of hardware and which does not operate concurrent with customer operations.

software, on-line maintenance -
Maintenance software designed for use in field maintenance of hardware and which operates under control of the operating system concurrent with customer operations.

subsystem -
An organized collection of hardware together with any necessary software, controlware, and/or firmware components operating within a system and performing functions assigned to it by the system. For example, a collection of tape devices, controller, controlware, and software devices is a magnetic tape subsystem. A processor and memory without the coded instructions necessary to process data would not be a subsystem.

system, computer -
An organized collection of interrelated software and hardware products, accessories and parts that are directly interconnected and contain only one mainframe under control of a single copy of an operating system and is designed to perform data processing functions.

systems, network of -
An organized connection of computer systems, software and hardware products, accessories and parts interconnected or interrelated in such a manner as to perform data processing functions.

system, operating -
Software which guides a processing system in the performance of its tasks by controlling the execution of computer programs and by providing support services to programs and programmers, e.g., scheduling, debugging, input-output control, etc.

TAR - Technical Action Request -
A report used by CDC hardware, firmware, and controlware users to report product design failures and errors. (Similar to PSR for software).

test, margin -
Test performed to provide information relative to a system's (unit's) ability to operate under the full range of design parameters. Normally accomplished by varying voltages and/or frequency.

test, product verification -
A test of a product or equipment in its operating (system) environment to determine that its operation, maintainability, and reliability meet the design criteria. The test includes the use of operating system and application programs, the use of maintenance procedures and diagnostic programs and operation over a prolonged period of time. The product verification test is generally performed on a preproduction or production unit.

testers, maintenance -
Equipment external to the system that initiates and performs fault detection and isolation and facilitates repair and adjustment.

time, active repair -
The interval during which activities occur at the user's location that are associated with implementing corrective or avoidance actions. Only those activities required to return the system or products to an operational state following the failure are included. Sometimes referred to as unscheduled repair time. For software this includes such things as dumping files, analysis, PSR documentation, installing corrective code, etc. The deferred installation of corrective code or PSR is considered scheduled maintenance.

time, actual usage -
The interval or accumulation of intervals during which an item is performing one or more of its intended functions.

time, administrative and logistic wait -
The interval during which support personnel or materials are not available.

time, calendar -
Calendar time is the elapsed interval of time during the measurement period, expressed in hours, day or months, as appropriate.

time, down -
The sum of active repair time, analysis time and administrative and logistic wait time which takes the system down during scheduled operating time. The interval during scheduled operating time when the item is inoperative.

time, analysis -
The interval the user spends determining that maintenance service is required.

time, lost -
The effective time that a system is in a totally unacceptable state for productive work as a result of interruptions. Lost time is the actual time lost to the user due to total or partial loss of system processing capability plus compensatory time for any reprocessing necessitated by interruptions. The following times, if present shall be included in the lost time calculation.

- analysis time
- administrative and logistics wait time
- active repair time
- reprocessing time

Under degraded conditions, elapsed time does not represent lost time since some useful work was processed during this interval. Lost time due to degradation is the product of the degradation factor and the time the system was in that degraded condition -- i.e., the time between the detection of an interruption and the point in time the users work has been restored to the state it would have been, had the interruption not occurred. NOTE: time in a degraded condition need not be contiguous -- e.g., re-run may be delayed resulting in a fully acceptable productive state between the interruption and the commencing of re-run.

The interval that service personnel are not allowed access to the item needing service is not included in the lost time calculation. The effect of system recovered errors on system thruput is not included in lost time.

time, reprocessing -
The sum of restoration time and rerun time.

time, rerun -
The amount of time required to return work in process to the state that it should have been in when the failure which caused the interruption or erroneous result was detected.

time, restoration -
The amount of time required to restore the operating system, and auxiliary subsystems, because of an interruption or erroneous result. The following activities, if present, shall be included in restoration time.

- restoration of remote devices
- system reinitialization

time, scheduled maintenance -
The dedicated system time to perform preventive and remedial maintenance and to install corrective PSRs and FCOs. Where this scheduled maintenance activity is performed concurrent, it is handled like degraded lost time with the use of a degradation factor. The installation of new features or products is not considered scheduled maintenance time.

time, scheduled operating -
The interval allocated in advance for the system to be operational for the user.

unit, accounting -
A single item in which the resources used by a job or required for a terminal session are combined including memory field length, CPU time, mass storage usage, magnetic tape usage, permanent file usage and unit record usage. The accounting unit of the NOS operating system is the System Resource Unit (SRU) and its specific definition is contained in CDC document 60435700 "NOS Installation Handbook".

Appendix B

Reliability Module Failure Rates

		Qty.	Failure Rate $(x10^{-6})$	Total $(x10^{-6})$
I.	LSI Device Module			
	LSI Array	1	0.087	0.087
	Terminators	9	0.001	0.009
	Ceramic Capacitors	2	0.002	0.004
	LSI Connector (52 pin)	1	0.104	0.104
	Solder Joints	4	0.0003	0.0012
			TOTAL -->	0.205
II.	LSI Half Pack Module			
	Half Pack	1	0.0412	0.0412
	Terminators	3	0.001	0.003
	Ceramic Capacitor	1	0.002	0.002
	Half Pack Connector (26 pin)	1	0.048	0.048
	Solder Joints	2	0.0003	0.0006
			TOTAL -->	0.0948
III.	LSI Board Module			
	Coax Connections	1,260	0.0028	3.528
	Vias	18,500	0.00005	0.925
	Ceramic Capacitors	330	0.002	0.660
	Solder Joints	660	0.0003	0.198
			TOTAL -->	5.311
IV.	F100K Module			
	F100K Device	1	0.0240	0.024
	Terminators	3	0.001	0.003
	Vias	30	0.00005	0.0015
	Solder Joints	24	0.0003	0.0072
			TOTAL -->	0.0357
V.	Average Auxiliary Board Module			
	Edge Connectors	338	0.002	0.676
	Ceramic Capacitors	150	0.002	0.30
	Solder Joints	300	0.0003	0.09
	F100K/Board (weighted avg.)	45	0.0357	1.607
			TOTAL -->	2.673

		Qty.	Failure Rate $(\times 10^{-6})$	Total $(\times 10^{-6})$
VI.	Bus Board Assembly			
	Tantalum Capacitors	150	0.014	2.10
	Solder Joints	300	0.0003	0.09
			TOTAL -->	2.19
VII.	RAM Module (4096-bit ECL)			
	RAM Device	1	0.07	0.07
	Solder Joints	18	0.0003	0.0054
	Vias	22	0.00005	0.0011
	Connector Pins	0.94	0.002	0.0019
			TOTAL -->	0.0784

Appendix C

Component Elemental Failure Rates

Used in FMP Evaluation

Semiconductors	Failures per 10^6 hrs.
LSI Array	0.087
F100K Array	0.024
ECL RAM	0.07
MOS RAM (65K)	0.463*
CCD Array	0.926**
TTL SSI	0.02
Other Components	
Ceramic Capacitor	0.002
Tantulum Capacitor	0.014
Terminators	0.001
Vias (Printed Wiring Boards)	0.00005
Solder Joints	0.0003
LSI Connector	0.104
Edge Board Connection	0.002/pin
Zero Insertion Force Connector	0.113
Coaxial Connector	0.0028

---

* MOS failure rate is based upon MIL-HDBK-217B and CDC experience. ** Since no reliability information is yet available for CCD storage circuits, the failure rate is estimated to be twice that of the MOS RAM devices.

Appendix D

COMPONENT PIECE PART FAILURE RATES

CONTROL DATA CORPORATION

SYSTEM STANDARD

STD 1.12.020
REV A
DATE August 1978
PAGE 1 of 12

PREDICTING RELIABILITY, AVAILABILITY AND MAINTAINABILITY PARAMETER VALUES IN HARDWARE AND SOFTWARE

1.0 SCOPE

1.1 Purpose - This standard defines methods of establishing Reliability, Availability and Maintainability (RAM) parameter values required by CDC Std 1.12.006 - Specifying and Measuring RAM, by using reliability prediction techniques. The use of the types of predictions described will provide a consistency and commonality for evaluating the predicted RAM performance of a product during its evolution. Use of the rates and factors contained in Appendices A and B will provide a common base of design information for performing MTBF predictions.

1.2 Applicability - This standard applies to all products intended to be offered for sale or lease by CDC unless specifically excluded by customer contractual conditions. The use of "products" in this document refers to modules, equipments, software, products, subsystems and systems.

1.3 Effectivity - This standard is effective immediately upon its release. This standard supersedes Standard Bulletin D003.

1.4 Authority - The enforcement of this standard is in accordance with CDC Policy 10:04:00. The interpreting authority for this standard is the General Manager, CDC Technical Standards.

2.0 APPLICABLE DOCUMENTS

2.1 Referenced Documents

CDC-POLICY 10:04:00 - CDC Technical Standards
CDC-STD 1.12.006 - Specifying and Measuring RAM (Not yet released)
CDC-STD 1.12.999 - Glossary of RAM Terms (Not yet released)
MIL-HDBK-217B - Reliability Prediction of Electronic Equipment
MIL-STD 756A - Reliability Prediction
CDC-Tech Memo 19 - Reliability Growth Prediction Procedure
Proceedings of 1968 Symposium on Reliability

2.2 Related Documents

CDC-Tech Memo 6 - Reliability Goals
MIL-HDBK-472 - Maintainability Prediction
CDC Pub-60435200 - Investment Decision Model

3.0 GLOSSARY

Refer to CDC-STD 1.12.999 - Glossary of RAM Terms.

4.0 REQUIREMENTS

Expected RAM parameter values, when specified in the following controlling documents, shall be determined (predicted) using the types of predictions and failure rates established by this standard (See Figure 1). Also, where predictions are required as a part of a design evaluation or system evaluation, the types established by this standard shall be used so consistency is maintained in comparative situations. Note: Design and Release predictions are only applicable to hardware MTBF predictions. (See Expository Remarks no. 2 and no. 3)

Approved by D.L. Bickel
D.L. Bickel, Vice President
Operations Services, Computer Group

Type of RAM Prediction \ Controlling Document	Strategy Documents or equivalent	Market Requirements Documents or equivalent	Design Objectives Document or equivalent	Design Requirements Documents or equivalent	Engineering Specifications or equivalent	Certification or Equivalent
Market Analysis	X	X	X			
Extrapolated			X	X	X	
Allocated			X			
Design				X	X	
Release						X

Figure 1 - TYPES OF RAM PREDICTION ASSOCIATED WITH VARIOUS CONTROLLING DOCUMENTS

4.1 Strategy and Marketing Documents - Reliability parameters to be specified in Strategy and Marketing Requirement Documents shall be based on a Market Analysis. (See 6.1)

4.2 Design Objectives Documents - Reliability parameters to be specified in Design Objectives Documents shall be based on a tradeoff between capabilities of the design and the market needs as expressed in the strategies and marketing requirements. The RAM values specified in Design Objective documents will give priority to market needs versus capabilities of design. (See 6.1, 6.2, and 6.3)

4.3 Design Requirements Documents - RAM parameters to be specified in Design Requirements Documents shall be based on the design and/or extrapolated predictions (see 6.2 and 6.4).

4.4 Engineering Specifications Documents - RAM parameters to be specified in Engineering Specification documents shall be design or release predictions. Should the design or release RAM parameter values not meet the Design Requirement values (see 6.2 and 6.4), a decision must be made to either hold the manufacturing release and continue the design activity or change the Design Requirement values to agree with the release values by formal DR revision and approval.

4.5 Certification - MTBF predictions used for validating hardware for release shall be Release Predictions. (See 6.5)

5.0 RESPONSIBILITIES

As with other standards, responsibility for implementation and enforcement rests with division management. Responsibility for updating failure rates and application factors is defined in the last paragraph of the Preface to Appendix A.

6.0 PROCEDURE

6.1 Market Analysis - The market analysis approach of establishing a RAM requirement is based on an analysis of market need and competition. Analysis of CDC and competitive RAM performance trends and expected technological advancements are also to be used in this prediction.

6.2 Extrapolated Predictions - Extrapolated predictions are projections based on historical data on similar Control Data and competitive products. Known RAM data on similar or predecessor products and the growth characteristic of such data are used to extrapolate or "predict" the RAM values for the proposed product. The extrapolated prediction is not based on a compilation of component/part failure or repair rates.

6.3 Allocated Predictions - Allocated predictions are RAM requirements assigned to individual products to attain a desired overall system RAM. These types of predictions are suitable where the overall system RAM and some product RAM requirements are specified and the remaining product RAM requirements are to be determined.

6.4 Design Predictions - (Applicable only to hardware MTBF predictions) Design predictions are based upon a design strategy as represented by a reliability block diagram. Design predictions produce inherent MTBF values which must be translated into expected observed predictions by use of K factors. The specific K factor used and the rationale for its selection should be documented as part of the prediction. (See Expository Remark 1) The procedure, using MIL-STD 756A - Reliability Prediction, as a guideline for preparing a design prediction is as follows:

6.4.1 Product Definition - The product for which the prediction is being made is defined in terms of:

- functional and physical boundaries
- conditions which constitute failures
- conditions under which the product is to operate
- required maintenance conditions

6.4.2 Reliability Block Diagram and Reliability Model - From the descriptions of the product definition, above, a reliability block diagram is constructed. Each block of the diagram is identified and any assumptions and simplifications are clearly stated. A mathematical equation (model) is derived based upon the relationships described by the block diagram. (see 6.4.4 for applicable assumptions)

6.4.3 Computed MTBF - An estimate of the quantity of parts comprising each functional block of the Reliability Block Diagram is derived. Based upon the Component/Parts Failure Rate Table (Appendix A), an evaluation of the failure rate for each block is made. These functional block failure rates are used in the mathematical equation to compute the design prediction of the subject product.

6.4.4 Prediction Assumptions - For the purposes of this standard, failure rates of individual blocks or components/parts within a block are assumed to be constant with respect to time so that mean time between failure (MTBF) is the reciprocal of mean failure rate. By this assumption, infant mortality and wear out are excluded.

Unless otherwise stated in specific reliability predictions, mean failure rates of components/parts and their application environments are those identified in Appendices A and B of this standard.

Each device denoted by a block in a reliability block diagram is assumed to be independent from all other blocks with regards to probability of failure.

The inherent predictions assume the design, manufacturing and service is perfect and in accordance to all applicable standards and/or guidelines unless specifically stated otherwise.

6.5 <u>Release Predictions</u> - (Applicable only to hardware MTBF predictions) Release predictions are based upon the completed design. A release prediction is the most accurate prediction because it includes information from detailed schematics, detailed environmental data, firm parts selection, and detailed application data. Release predictions produce inherent MTBF values which must be translated into expected observed predictions by use of K factors. The specific K factor used and the rationale for its selection should be documented as part of the release prediction. (See Expository Remark 1)

The procedure using MIL-STD 756A - Reliability Prediction, as a guideline, for developing a release prediction is the same as for a design prediction (see 6.4) with the following additional considerations.

6.5.1 Product Definition - This definition will represent the actual configuration and include the conditions of manufacturing, shipping and handling, and maintenance and operating procedures.

6.5.2 Reliability Block Diagram and Reliability Model - The assumptions, constraints and simplications of the block diagram and mathematical model are based upon the detailed schematics and the conditions in the product definition. (See 6.4.4 for applicable assumptions)

6.5.3 Computed MTBF - The types and quantities of parts comprising each block are derived from the detailed parts list of the product. Failure rates for the components are listed in the Component/Parts Failure Rate Table (Appendix A). State the failure rates used when the component/parts are not listed in Appendix A or where stress factors other than the nominal stress environment (Appendix B) exist for particular components.

APPENDIX A

COMPONENT/PIECE PART FAILURE RATES

PREFACE

The failure rates listed in the following table reflect current capabilities of the individual components/piece parts under nominal stress levels. These failure rates are to be used in predictions as required and discussed in the prediction standard to which this appendix is attached. Prior to the use of the following tables, it is strongly recommended that the preface be read in full in order to obtain a clear understanding of the basis and underlying assumptions to the failure rate data.

Failure Definition/Units

When using the enclosed failure rates, note that the term "failure" is defined as an open, short, or parameter change greater than specified tolerance. These rates are based primarily on solid failures and do not necessarily include the effect of intermittents and transients. The failure rates are inherent failure rates for each generic part type. The term "inherent" is defined to mean the reliability that will be observed on a mature component in a mature application. Both the component and the application have had sufficient power-on time to have passed "infant mortality". When calculating observed failure rates to compare to these inherent numbers, the calculation should be done to a 60% confidence level.

The units of measurement for failure rate is "failures per million hrs."

Stress Levels

The inherent failure rates are defined for nominal stress conditions. Assumptions include a junction temperature of 45°C and unless otherwise specified all semiconductor packages are hermetically sealed. All components are assumed to be in a ground benign environment which is defined by nearly zero environmental stress with optimum engineering operation and maintenance.

Source of Data

The failure rate source codes are A-CDC data; B-Other Manufacturer's data; C-Component Industry data; D-Defense (MIL-HDBK-217B)/NASA; and E-Engineering judgment. They are listed in their order of precedence. The most accurate data applicable to the types of components and equipment that CDC uses and produces is, naturally, data from existing CDC equipment. Military data, generally being compiled from environments and equipment different from those of CDC and manufacturers of equipment similar to CDC's carries somewhat less weight. When no data exists to support an inherent failure rate for a component, an engineering judgment must be made. It is based on a reliability comparison with a component which has a known failure rate. Factors which influence this comparison include electrical complexity, power dissipation, technology employed with its associated strengths and weaknesses, and materials.

Updating Responsibilities

As a result of manufacturers continuously improving their products (components/piece parts) and users becoming more sophisticated in the application of those products, a continual change in the failure rates of those components/piece parts can be expected. In order to stay abreast of these changes, it's necessary to implement a mechanism for providing periodic updates to this appendix. The primary input for this mechanism will be the users of the data contained herein. All such users are strongly urged to submit recommended changes which they believe would improve the validity of the tables' contents. Such change should be sent to CDC Technical Standards in care of J. E. Mikkonen, HQW11H, with a discussion of the recommended changes and supportive data on the change. No more frequent than quarterly, all such changes will be reviewed by Reliability Engineers from various CDC divisions. At the completion of a successful review process, an updated failure rate table will be published and distributed.

COMPONENT/PART FAILURE RATES

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Section 1 Microcircuits			
ECL 10K SSI	.01	A/C	1/3/77
ECL 10K MSI	.01	A/C	1/3/77
ECL 10K Transmitters/ Receivers/ Interface Circuits	.02	A/C	1/3/77
ECL 10K			
10101	.0094	D	3/7/77
10102	.0094	D	3/7/77
10105	.0080	D	3/7/77
10107	.0080	D	3/7/77
10109	.0065	D	3/7/77
10110	.0065	D	3/7/77
10114	.0080	D	3/7/77
10117	.0094	D	3/7/77
10121	.0094	D	3/7/77
10124	.0094	D	3/7/77
10125	.0094	D	3/7/77
10129	.03	D	3/7/77
10130	.012	D	3/7/77
10131	.013	D	3/7/77
10133	.019	D	3/7/77
10135	.02	D	3/7/77
10136	.035	D	3/7/77
10141	.032	D	3/7/77
10145	.01	D	5/15/78
10160	.015	D	3/7/77
10161	.017	D	3/7/77
10164	.017	D	3/7/77
10165	.030	D	3/7/77
10166	.023	D	5/15/78
10173	.021	D	3/7/77
10176	.018	D	5/15/78
10179	.017	D	3/7/77
10181	.045	D	3/7/77
10192	.03	D	3/7/77
10800 Micro Processor	.15	D/E	11/10/77
10803 Interface Memory	.83	D/E	11/10/77
ECL 10K RAMS			
256 bits	.035	C/D	1/3/77
1024 bits	.07	C/D	1/3/77

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
ECL 100K			
100101	.0080	D	3/7/77
100102	.012	D	3/7/77
100107	.012	D	3/7/77
100112	.010	D	3/7/77
100114	.012	D	3/7/77
100117	.015	D	3/7/77
100130	.029	D	3/7/77
100150	.040	D	3/7/77
100151	.040	D	3/7/77
100155	.038	D	3/7/77
100160	.018	D	3/7/77
100170	.030	D	3/7/77
100171	.022	D	3/7/77
100181	.089	D	3/7/77
TTL SSI			
Ceramic	.02	A/C	
Novolac Plastic	.04	C	1/3/77
Non-Novolac Plastic	.07	B/C	1/3/77
TTL MSI			
Ceramic	.03	A/C	
Novolac Plastic	.06	C	1/3/77
Non-Novolac Plastic	.105	C	1/3/77
TTL Transmitters/ Receivers/ Interface Circuits			
Ceramic	.03	D	1/3/77
Novolac Plastic	.06	C	1/3/77
Non-Novolac Plastic	.105	C	1/3/77
TTL RAMS			
256 bit	.035	C/D	1/3/77
1024 bit	.07	C/D	1/3/77
TCS			
Ceramic	.01	A	4/12/76
Non-Novolac Plastic	.2	A	4/12/76
DTL			
Ceramic	.01	C	
Non-Novolac Plastic	.3	C	
MOS RAMS			
256 bit	.035	C/D	1/3/77
1024 bit	.07	A/C	1/3/77
4096 bit	.2	C/D	1/3/77
16384 bit	.444	A/D	5/15/78
LINEAR			
Op Amps	.05	E	
Sense Amps	.05	E	
Simple NPN Darlington AMP	.075	E	
Voltage Regulators	.07	E	
Digit Drivers	.05	E	

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
ROMS			
256 bit	.02	D	1/3/77
1024 bit	.047	D	1/3/77
Section 2 Discrete Semiconductors			
Silicon Transistors			
Logic NPN	.006	A	4/12/76
PNP	.009	A/D	1/3/77
Memory NPN	.04	A	4/12/76
PNP	.06	A/D	1/3/77
Power NPN	.1	C	4/12/76
PNP	.15	A/D	1/3/77
Dual Logic NPN	.01	E	4/12/76
Darlington	.08	D	12/1/77
Quad Memory NPN			
Novolac Plastic	.32	C	1/3/77
Ceramic	.16	A	1/3/77
u-T Au-Al White Cap	.006	A	4/12/76
u-T Au-Au Black Cap	.0015	A	1/3/77
u-T Au-Al Red Cap	.0015	A	4/12/76
u-T Au-Al Yellow Cap	.0015	A	4/12/76
u-T Au-Al Blue Cap	.0026	A	4/12/76
u-T Au-Al Green Cap	.0043	A	4/12/76
Germanium Transistors			
Logic NPN	.025	A/D	1/3/77
PNP	.009	A/D	1/3/77
Memory NPN	.2	A/D	1/3/77
PNP	.07	C	4/12/76
Power NPN	.8	A/D	1/3/77
PNP	.3	C	4/12/76
FET	.076	D	4/12/76
Silicon Diode			
Logic	.00024	A	
Germanium Diodes			
Logic	.0024	A	
Silicon Power Rectifier	.2	C	
Zener Diode	.027	D	4/12/76
SCR	.04	C	
Thyristor	.023	D	4/12/76
Thermistor	.01	C	
LED plastic	.2	C	4/12/76

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Section 3 Resistors			
Carbon Comp 1/8 w	.00045	A/D	4/12/76
1/4 w	.00002	A	
1/2 w	.0008	C/E	
1 w	.0008	C/E	
2 w	.004	C/E	
3 w	.004	C/E	
Wire Wound 1 w	.01	C/E	
2 w	.01	C/E	
3 w	.02	C/E	
5 w	.02	C/E	
Variable Wire Wound 1/2 w	.02	C/E	
1 w	.02	C/E	
2 w	.02	C/E	
3 w	.04	C/E	
5 w	.1	C/E	
Metal Film 1/8 w	.002	C/E	
1/4 w	.002	C/E	
1/2 w	.004	C/E	
1 w	.008	C/E	
Terminators Thin/Thick Film			
SIP-6 Resistors + 1 Capacitor	.017	D	4/12/76
DIP-12 Resistors + 2 Capacitors (TE2 & TE7 Type)	.022	A	4/12/76
DIP-12 Resistors + 2 Capacitors (R100 & R500 Type)	.014	A/D	4/12/76
Section 4 Capacitors			
Ceramic	.002	A	4/12/76
Electrolytic, Aluminum	.02	A	
Electrolytic, Paper	.1	A	
Mica dipped	.0003	D	4/12/76
Mica molded	.003	D	4/12/76
Mica button	.12	D	4/12/76
Mylar	.001	D	4/12/76
Paper/Plastic	.0002	D	4/12/76
Paper/Plastic	.0002	D	4/12/76
Tantalum	.014	A	4/12/76
Variable Air	.1	D	4/12/76
Section 5 Inductive Devices			
Inductors/Chokes	.018	A	4/12/76

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Pulse Transformers			
Discrete	.01	A	4/12/76
Encapsulated	.005/core	A	4/12/76
Hybrid (TR00 Type)	.02	D	4/12/76
Power Transformers	.0075	D	4/12/76
RF Transformers	.0096	D	4/12/76
Contactors	.11	A/E	5/15/78
Relays, General Purpose	.13	D	4/12/76
Motors	1.0	D	4/12/76

Section 6 Connectors/Connections

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Edgeboard Connector			
Mainframe Environment	.002 per pin	A	1/3/77
Peripheral Environment	.006 per pin	A	1/3/77
PC board conn. (3500 style)	.0036 per pin	A	1/3/77
PC board conn. (6000 style)	.002 per pin	A	1/3/77
PC board conn. (7000 style)	.0017 per pin	E	1/3/77
Conn. Pins, Cable Connector	.00013 per pin	A	
Coax Connector (includes inner and outer contact)	.0014	C/E	1/3/77
Power Connector	.002 per pin	E	1/3/77
DIP Sockets, gold			
single contact	.003 per pin	E	1/3/77
dual contact	.002 per pin	A/E	1/3/77
Multiple contact	.001 per pin	E	1/3/77
Solder Joints			
Plated thru hole	.00015	A/D	1/3/77
Surface/lap	.0003	A	1/3/77
Non-plated thru hole	.00044	D	1/3/77
Other hand solder	.0044	D	1/3/77
Taper Pins	.00017	A	1/3/77
Wire Wraps	.0000037	D	1/3/77
Crimp Joints	.000132		

Section 7 Refrigeration and Cooling

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Regulator, Hot Gas Bypass	2.650	A/E	5/15/78
Valve, Water Regulating	2.650	A/E	5/15/78
Valve, Expansion	.589	A/E	5/15/78
Valve, Angle, Refrigeration			
Valve, Solenoid (MB1452)	1.990	A/E	5/15/78
Valve, Solenoid (MB952)	1.990	A/E	5/15/78
Condenser	2.650	A/E	5/15/78
Compressor, 2-Ton	1.330	A/E	5/15/78
Filter, Drier	.300	C	5/15/78
Fitting, Fusible Half-Union	2.650	A/E	5/15/78
Gauge, Pressure	1.300	A/E	5/15/78
Control, Dual Pressure	.320	C	5/15/78
Eliminator, Vibration	.039	C	5/15/78
Joints, Flare	.040	C	
Joints, Threaded	.040	C	
Quick Disconnect	.800	C	

Component/Part Description	Failures Per Million Hours	Source Code	Change Date
Thermostat	.250	C	
Thermistor	.600	C	
Distributor	.140	C	
Condensing Unit (2-Ton)	20.200	C/A/E	
Condensing Unit (3-Ton)	20.200	C/A/E	
Condensing Unit (5-Ton)	20.200	C/A/E	
Blower Assembly	6.0	D	4/12/76
Muffin Fan	2.4	E	4/12/76
Section 8 Miscellaneous			
Circuit Breakers	.16	C	4/12/76
Delay Lines	.054	A	
Fuses	.1	D	4/12/76
Lamps, Neon	.2	D	4/12/76
Lamps, Incandescent	1.0	C/D	4/12/76
Memory Cores	.0000014	A	4/12/76
Switches, Rotary	.42	D	4/12/76
Switches, Toggle	.17	D	4/12/76
Quartz Crystals	.2	D	4/12/76
Wire Jumpers	.00013	A	
Printed Wiring Boards			
Multilayer	.00005 per plated thru hole	A	1/3/77
Two sided	.000006 per plated thru hole	D	1/3/77
Multiwire (thin)	.0018 per hole		12/1/77
Multiwire (thick)	.0107 per hole		12/1/77

APPENDIX B

COMPONENT APPLICATION FACTORS

The standard component failure rates in Appendix A are established in consideration of standard application environments. Application in either a more relaxed or more severe operating environment normally will affect the failure rate. The presentation of this data is that the majority of components in any given equipment will be operated at or below these conditions. New input to keep this data current with the state of the art is solicited for consideration in future revisions.

General - Cooling Air Temperature: 25°C

Semiconductors	
Integrated Circuits	
Operating Junction Temperature	45°C
Operating Voltage - Digital Circuits	±5% of mfg. nominal
- Linear Circuits	75% of maximum rating
Transistors, Silicon	
Operating Junction Temperature	25°C below maximum rating
Voltage and Current Ratings	75% of maximum rating
Diodes & Rectifiers, Silicon	
Operating Junction Temperature	25°C below maximum rating
Voltage and Current Ratings	75% of maximum rating
Resistors	
Carbon Composition; Carbon Film; Metal Film; and Wire Wound, Power	
Power Dissipation	50% of maximum rating
Operating Voltage	75% of maximum rating
Capacitors	
Ceramic, Glass, Paper, Mylar, Mica	
Operating Voltage	75% of maximum rating
Electrolytic	
Operating Voltage	90% of working voltage
Ripple Current Effect	maximum temperature rise of 15°C above ambient
Tantalum	
Operating Voltage	75% of working voltage
Transformers & Inductors	
Operating Temperature	15°C below maximum insulation rating
Voltage Rating	75% of working dielectric rating
Fuses	
Operating Current	75% of nominal rating
Switches & Relay Contacts (other than dry circuit conditions)	75% of nominal rating

Appendix E

FMP Logic Unit Elemental Failure Rates

I. Scalar

Modules	Quantity		Failure Rate
Auxiliary Boards	45		121.19
LSI Boards	16		84.98
LSI Arrays	1967		403.24
Half Packs	170		16.12
Clock Oscillator	1		4.97
Bus Board Assemblies	16		35.04
		TOTAL -->	665.54

II. Swap

Modules	Quantity		Failure Rate
LSI Boards	2		10.62
LSI Arrays	272		55.76
Bus Board Assemblies	2		4.38
		TOTAL -->	70.76

III. Intermediate Map

Modules	Quantity		Failure Rate
Auxiliary Boards	10		26.73
LSI Boards	2		10.62
LSI Arrays	204		41.82
Bus Board Assemblies	2		4.38
		TOTAL -->	83.55

IV. Main Map

Modules	Quantity		Failure Rate
Auxiliary Board	10		26.73
LSI Boards	3		15.93
LSI Arrays	304		69.70
Bus Board Assemblies	3		6.57
		TOTAL -->	118.93

V. Memory Interchange

Modules	Quantity		Failure Rate
LSI Boards	12		63.73
LSI Arrays	1632		334.56
Bus Board Assemblies	12		26.28
		TOTAL -->	424.57

VI. Vector Streaming

Modules	Quantity	Failure Rate
Auxiliary Boards	10	26.73
LSI Boards	4	21.25
LSI Arrays	476	97.58
Bus Board Assemblies	4	8.76

TOTAL --> 154.32

VII. Vector

Modules	Quantity	Failure Rate
Auxiliary Boards	200	534.6
LSI Boards	45	239.00
LSI Arrays	4760	975.80
Bus Board Assemblies	45	98.55

TOTAL --> 1847.95

VIII. Streaming Control

Modules	Quantity	Failure Rate
Auxiliary Boards	20	53.46
LSI Boards	1	5.31
Bus Board Assembly	1	2.19

TOTAL --> 60.96

IX. I/O

Modules	Quantity	Failure Rate
Auxiliary Boards	20	53.46
LSI Boards	4	21.25
LSI Arrays	408	83.64
Bus Board Assemblies	4	8.76

TOTAL --> 167.11

Appendix F

Unit Functional Failure Rate

A unit which has fault correcting capability will have a functional failure rate different from its elemental failure rate (See appendix A, CDC-STD 1.12.999 *Glossary of Reliability, Availability, and Maintainability Terms*).

The functional failure rate for the unit will be the sum of the elemental failure rate of that portion not included within the fault correcting part plus the functional failure rate of the fault correcting part.

To determine a unit's functional failure rate, the elemental failure rate is first computed by summing the products of the part type failure rates times the number of parts of each type (see appendix E). From this is subtracted the elemental failure rate of the fault correcting part of the unit. To the remainder is added the functional failure rate of the fault correcting part as determined by the SECDED model. Three examples of these computations are given here. It should be noted that scheduled maintenance is assumed to be once per week for all units except Backing Store, which is assumed to be once per day.

Backing Store Unit (262K-bit array)

Component	Quantity	Failure rate	
		Unit	Total
Storage Device	36,864	.926	34,136.06
Storage Device Connector	36,864	.048	1,769.47
TTL Support Circuits	14,400	.02	288.00
Board Connectors	144	.284	40.90
Capacitors	11,520	.014	161.28
Solder joints	1,107,856	.00015	166.18
Vias	1,254,628	.00005	62.73
Unit Total Elemental Failure Rate			36,624.62

The configuration of the Backing Store is four data bits per replaceable module (board) and it is so arranged that each data bit is in a different SECDED sector. Because of this arrangement, the total failure rate of the board is divided into four parts, each essentially totally corrected by SECDED. The elemental data bit failure rate used in the SECDED model is the total elemental failure rate divided by four times the number of boards (144) in the memory. This yields a SECDED sector elemental data-bit failure rate of 63.58 failures per 10^6 hours.

The SECDED arrangement of the memory is eight sectors in parallel, each sector having a single rank of 72 data bits. The SECDED model derives a failure rate of 230.7 so the total Backing Store has a functional failure rate of 8 times this or 1845.6 and a functional MTBF of 541.8 hrs.

Main Memory Unit

Component	Quantity	Failure rate Unit	Failure rate Total
Storage Device	159,744	.078	12,380.2
F100K Device	29,120	.0357	1,039.6
Memory Interface	(1)		302.2
Cabinet	(2)		576.5
Elemental Failure Rate			14,298.5
Less Storage Elemental Failure Rate			12,380.2
Remainder			1,918.3
Storage Functional Failure Rate			6.0
Main Memory Functional Failure Rate			1924.3
Main Memory Functional MTBF			519.67

The Main Memory Unit will utilize the CYBER 203 memory interface with new storage units which are now in development. Therefore all but the storage device elements are estimates based on preliminary design configurations.

1) The failure rate for the interface unit is that determined for the CYBER 203 (254×10^{-6}) plus the failure rate of 156 extra LSI devices (48.2×10^{-6}).

2) The failure rate for the cabinet (power connections, filter capabilities, etc.) is twice that of the CYBER 203 since the Main Memory will use twice the number of storage devices as for a 1 million word CYBER 203 using 1K chips. (As mentioned before, a new memory chassis is currently being designed, thus the inability to count the expected number of memory chassis for the FMP.)

Transfer Units (Main Map, Memory Interchange, Vector Streaming)

Main Map Unit elemental failure rate	118.3
Memory Interchange elemental failure rate	424.6
Vector Streaming Unit elemental failure rate	153.7
Total elemental failure rate	696.6

The transfer path elemental failure rate is based upon the following assumptions:

1) The transfer path is made up of 14 devices in series.

2) There are 6 parallel data transfer bits per device.

3) 0.8 of a device comprise the transfer paths; the other 0.2 is in control logic (not corrected by SECDED).

4) There are 16 39-bit SECDED units comprising the total transfer path (512 information bits).

The equivalent failure rate of a transfer bit within a device is one sixth of 0.8 of the LSI device module failure rate plus the proportionate failure rate of coax connections and vias. (See appendix B, LSI Board Module. Use the failure rate of coax and vias divided by 150 -- the number of LSI devices per board.)

device bit failure rate = (0.205 + 0.030)0.8/6 = 0.0313

The total bit transfer path failure rate is 14 times this value = 0.439. (This value is used in the SECDED model.) There are 39 bits in a SECDED sector, and 16 sectors make up the width of the transfer path.

Transfer path total elemental failure rate = 0.439 x 39 x 16 = 273.7

The functional failure rate of the transfer path is calculated from the SECDED model with one rank (n = 1) and an elemental failure rate of 0.439. This calculation yields a failure rate of 2.73.

The Transfer Units functional failure rate is:

Total elemental failure rate	696.6
Less the Transfer Path elemental failure rate	- 273.7
Remainder	422.9
Plus the Transfer Path functional failure rate	+ 2.7
Transfer Path functional failure rate	425.3

In a similar manner, the functional failure rates of the other transfer paths are computed using the following assumptions:

1) The Swap Unit transfer path is 512 data bits wide and is composed of eight 72-bit (including check bits) wide SECDED units in parallel. The SECDED unit is three LSI devices deep (that is, the transfer path has three devices in series) but for model computation purposes it is treated as a single rank with each device having a failure rate of three times that of a single LSI device.

2) The Intermediate Map Unit transfer path is 256 data bits wide and is composed of four 72-bit wide SECDED units in parallel. The transfer path has three devices in series but for model computation purposes it is treated as a single rank with each device having a failure rate of three times that of a single LSI device.

3) A translation from a 72-bit SECDED sector to two 39-bit SECDED sectors, and vice versa, is considered to take place at the interface of the Intermediate Map Unit and the Main Map Unit. The translation is accomplished by check and generation circuits.

Appendix G

The SECDED Model

The model for computing the functional failure rate of a SECDED unit is developed from the basic reliability formulas:

$$1 = R + Q \quad \text{and} \quad R = e^{-\lambda t}$$

where R is the probability of success, that is, the probability of no failures; Q is the probability of encountering a failure; λ is the failure rate of an element; and t is the time interval in question. The probability of success or failure for a rank of c elements is

$$(R+Q)^{c} = R^{c} + cR^{(c-1)}Q + \frac{c(c-1)\,R^{(c-2)}Q^{2}}{2!} + \ldots + Q^{c}$$

Since the first term is the probability of no failures occurring and the second term is the probability of exactly one failure occurring (which is correctable and therefore not a functional failure), the probability of no functional failures in the rank of elements within a SECDED sector of c bits is

$$R^{c} + cR^{(c-1)}Q$$

The probability of no functional failures occurring within a SECDED unit of n ranks is

$$P = (R^{c} + cR^{(c-1)}Q)^{n}$$

This equation is solved arithmetically with the values for λ and t (in the equations $R = e^{-\lambda t}$ and $Q = 1 - R$) set to the failure rate of the component and the maintenance interval, respectively.

If P is the probability of no failures in a SECDED unit and F is the functional failure rate of the unit, then

$$P = e^{-Ft} = 1 - Ft \text{ (for } Ft<0.05\text{) and}$$

$$F = \frac{1 - P}{t}$$

(Throughout this study the worst case condition of a whole chip failing has been used for memory failures since the predominant modes of partial chip failure are not conclusively known. If it is desired to consider partial chip failures, the component failure rate should be multiplied by the value of the average or major mode of partial chip failure and the number of ranks divided by that value.)

Appendix H

SYSTEM FUNCTIONAL AVAILABLITY AND RELIABILITY

The functional availablity-reliability of the NASF system can be determined for a user if the following use and system parameters are known.

1. Run or use time of the user program.

2. The system components required by the user program and the portion of time the program is "in" each component.

3. The amount of time of system overhead (e.g. operating system or controlware) for each of the programmable components.

4. The functional failure rate and mean down time (MDT) for each system component. The MDT for a non-redundant component is its MTTR. The MDT for a redundant component is the switch time (between the redundant components), if the switch time is extremely small compared to the component's MTBF.

An example is developed here to show how the above information is reduced to an availability and a reliablity figure for a user. Figure 1 is the reliability configuration for a given user job or task and table 1 shows the values for the components.

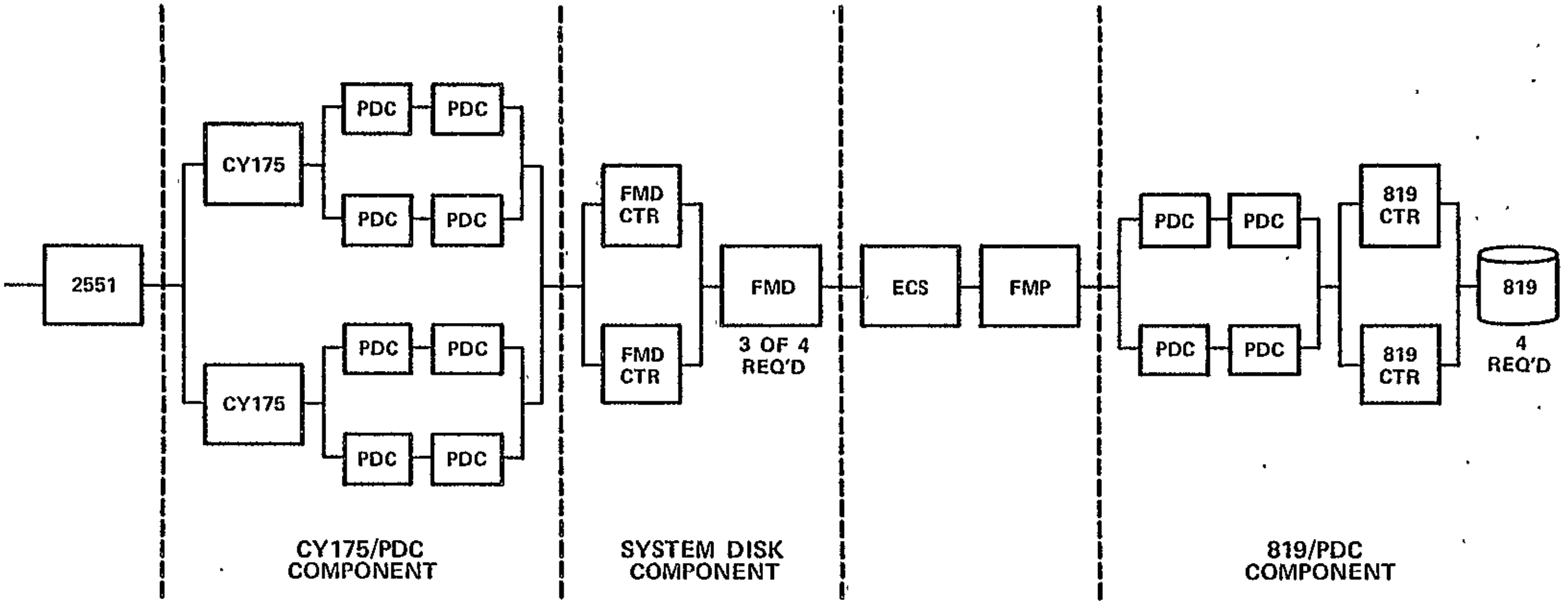


Figure 1. Reliability Model Configuration

Table 1
System Component Parameters

System Component	Functional Failure Rate	User Time	Overhead	(AFR) Applicable Failure Rate(1)	Mean Down Time
2551	541.7	0.1	0.1	108	1.6
CYBER 175	2725	0.3	0.25	1500	0.2(3)
PDC	100	0.2(2) 0.1(2)	0.1	30(2) 20(2)	0.1(3)
FMD	200	0.15	0	30	0.25(3)
FMD Controller	125	0.15	0	19	0.1(3)
ECS	1587.3	0.15	0	238	1.8
FMP	9373.8	0.65	0.05	6565	1.5
819 Disk	357.1	0.1	0	36	2.2
819 Controller	100(4)	0.1	0.1	10	0.1(3)

NOTES:

(1) Applicable failure rate is the system component functional failure rate times the sum of the user time and overhead.

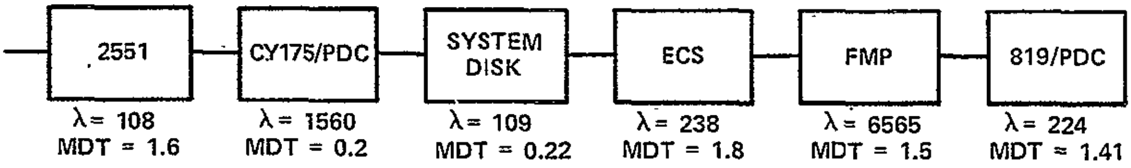
(2) PDCs associated with the 175 have a 0.2 user time factor. Those with the 819s have 0.1 user time factor.

(3) Mean down time is switch time between redundant components.

(4) The total failure rate of the dual controller is divided evenly between the two halves.

Further assumptions regarding the use of the system are:

1. Three of the four FMD's are required for system operation.
2. The probability of a failure in the tape and system disk subsystems is negligible (0.0004) during the loading of an alternate system disk required upon a disk failure (a ten minute period of time).
3. The 819 disks have no back-up (for this particular case).
4. The PDC networks associated with the 175s and the 819 subsystem are simplified (for ease of computation) into the configurations shown in figure 2.
5. Intuitively, it can be shown that the functional failure rate of redundant components with very short switch times is the same as the failure rate of one of the components. (A rigorous proof exists.)
6. The user time required is two hours.



Note: λ is the failure rate for a system component.

Figure 2. Simplified Reliability Configuration

The Simplified Reliability Configuration (figure 2) is derived in the following manner:

175/PDC Component

PDC failure rate = 2 times the applicable failure
                   rate (AFR) = 2x30 = 60
PDC MDT   = 0.1

(for the network of 4 PDC's in figure 1 - see assumption 5)

```
              175           PDC Network
          ----------        ---------
     _____|AFR=1500|________|AFR=60  |
          |MDT=0.2 |        |MDT=0.1|
          ----------        ---------
```

175/PDC failure rate = 1500 + 60 = 1560

$$\text{175/PDC MDT} = \frac{\Sigma(\text{AFR})(\text{MDT})}{\Sigma\,\text{AFR}} = \frac{0.2\text{x}1500+0.1\text{x}60}{1560} = 0.196$$

System Disk Component

```
          -----  ------   ------   ------
     -----|Ctr|---|Disk|---|Disk|---|Disk|
        |  -----  | ------   ------   ------
        |         |
        |  -----  |
       --|Ctr|--
           -----
```

Controller failure rate = 19
Controller MDT   = 0.1

Disk failure rate = 3 times AFR = 3x30 = 90
Disk MDT   = 0.25

System Disk Failure rate = 19 + 90 = 109

$$\text{System Disk MDT} = \frac{\Sigma(\text{AFR})(\text{MDT})}{\Sigma\,\text{AFR}} = \frac{19\text{x}0.1+90\text{x}0.25}{109} = 0.22$$

819 Disk Component

819 failure rate = 36
819 MDT   = 2.2

Controller (1/2) failure rate = 10
Controller MDT   = 0.1

Controller/819 failure rate = 36 + 10 = 46

$$\text{Controller/819 MDT} = \frac{\Sigma(AFR)(MDT)}{\Sigma AFR} = \frac{36 \times 2.2 + 10 \times 0.1}{46} = 1.74$$

Four controller/819 units have failure rate of 4x46=184
and MDT =1.74

PDC failure rate = 2 x AFR = 2 x 20 = 40
PDC MDT = 0.1

819 Disk failure rate = 184 +40 = 224

$$\text{819 Disk MDT} = \frac{\Sigma(AFR)(MDT)}{\Sigma AFR} = \frac{184 \times 1.74 + 40 \times 0.1}{224} = 1.41$$

NASF user availability-reliability is derived from the overall failure rate and MDT for the system for the user job configuration. The pertinent relationships are:

1. Failure rate (λ) of the system is the sum of component failure rates (see Reduced Reliability Configuration).

   $\lambda = \Sigma AFR$ (of each component)

2. System unavailability is the sum of each component unavailability which is the product of each component's failure rate and MDT.

   $\lambda MDT = \Sigma(AFR)(MDT)$ (of each component)

3. System availability is 1 minus the system unavailability.

   $A = 1 - \lambda MDT$ (of system)

4. The system reliablity (for very small λt) is

   $R = 1 - \lambda t$, where t = user required time.

For this example:

System $\lambda = 8804 \times 10^{-6}$ failures per hour (MTBI = 113.6 hrs)
System λ MDT = 0.0111
System Availability = 0.9889 or 98.89%
System Reliability = 1 - 0.0088 x 2 = 0.9824 or 98.24%

The operating system critical MTBF is derived on the assumption that the operating system works non-concurrently 25% of the time in the 175s and 5% of the time in the FMP, and that ECS must be operable during this time.

0.25 x 2725.0 = 681.25
0.05 x 9373.8 = 468.69
0.30 x 1587.3 = 476.19
O.S. Critical Failure rate 1626.13

O.S. Critical MTBF = 615 hours = 3.7 weeks

DIVISION 7

MAINTENANCE STUDY FOR THE NUMERICAL AERODYNAMIC SIMULATION FACILITY

DIVISION 7

MAINTENANCE STUDY FOR THE NUMERICAL AERODYNAMIC SIMULATION FACILITY

STRATEGY ASSUMPTIONS

The maintenance strategy for the NASF system is based on the assumption that there are two categories of equipment: system critical and system non-critical. System critical devices are those which must be operational before useful work can be accomplished. They consist of the FMP, the Network Processors (2551-1), and the ECS. System non-critical devices are those redundant equipments that need not all be operational before useful work can be accomplished. They consist of all equipments except those processors listed above.

Equipment was designated system critical based on how much the loss of its function would impair the system's usefulness. Without the FMP the system could continue to cue jobs and do support processing activities, however, no jobs could be run and the system's useful output would rapidly diminish. Loss of a network processor in the system would mean half the interactive users could not access their data base. ECS is system critical because standard software for the Support Processing System (SPS) depends on this equipment to coordinate the two SPS processors.

System non-critical equipment like the two CYBER-175 processors are each capable of doing the entire SPS task during a temporary interruption in one processor or the other. Mass storage subsystems (disk and cartridge) each have redundant capabilities which eliminate the possibility of a single interruption disabling a significant portion of the system.

The operating software developed for the NASF system has to support this categorization of the hardware configuration to minimize system interruptions. Failsoft and reconfiguration capabilities need to be an integral part of the software in order to take advantage of the hardware system redundancy.

These conditions will enhance the operation and maintenance of the system.

FIELD ORGANIZATION

The Engineering Services field organization will be operating out of a local service center by the time this system is installed. This type of organization allows efficient distribution of mobile service personnel among local installations. The service options under this organization vary from totally on-call, where the customer engineer (CE) is called when needed, to on-site coverage, where CEs are assigned to one installation.

Examination of the maintenance activity required for the NASF system indicates that optimum service can be obtained by assigning CEs to provide immediate response to interruptions which cause the system to be down during the normal work week (24 hours per day, Monday-Friday). This will cover 70% of the system down interruptions. The remaining 30% and all interruptions which do not cause the system to be down can be handled on-call from a local service center.

Response time for on-call service typically averages two hours or less. All of the PM and scheduled maintenance actions would be handled by the service center.

Estimated system maintenance cost for this option is $80,000 per month. Other options are available which would increase the immediate response to interruptions from 70% to 100%. This option would increase the cost by approximately 40% to $112,000 per month. Another option would decrease the immediate response to interruption capability to 50% and decrease the cost to approximately $70,000 per month. Initial spare parts cost for the total NASF are estimated to be $175,000 including $115,000 for initial FMP spare parts.

PREVENTIVE MAINTENANCE

A program of preventive maintenance (PM) will be implemented to minimize system interruptions. There are two categories of PM: dedicated and concurrent. Dedicated PM implies that a significant portion of the system will be used for this purpose and other useful work is not practical during this period. Concurrent implies that PM will be performed while the rest of the system is doing useful work.

Weekly dedicated PM is expected for the FMP. During this period single solid Main Memory failures may be removed and diagnostics will be run with margins on the rest of the CPU. This is expected to take four hours per week.

Daily concurrent PM requiring less than an hour is expected for the FMP. This will be required to remove single solid failures from the Intermediate Memory and Backing Store.

PM on all other equipment will be performed on a unit basis concurrently with system operation. This requires that the system be reconfigured so it doesn't use the equipment on which PM is being performed. For the CYBER-175 equipment a 3 hour period of concurrent PM will be required weekly for each unit.

It is expected that Intermediate Memory and Backing Store solid failures will be repaired concurrently with system operation. This will require reconfiguring the memory so the system doesn't use the portion (512K) being repaired.

PM tasks such as periodic replacement of filters and measurement of voltages and waveforms will be performed concurrently without affecting the operation of the system.

COMPUTER AIDED MAINTENANCE

All PM will be scheduled with the aid of a Computer Aided Maintenance Scheduler (CAMS) program. CAMS uses system configuration and error log informaton to optimize the PM schedule. This program can be run on any CYBER 170 system.

An error log for the FMP, the PDCs and the 819s will be maintained on the Maintenance Control Unit (MCU) disk. The MCU will analyze this log and provide reports on errors that occurred during system operation, as well as diagnostic errors.

A similar log for the CYBER-175 systems and the peripherals attached to the SPS will be stored on one of the system FMD disk units. The CYBER-175s will provide error reports similar to the one provided by the MCU.

MAINTENANCE SOFTWARE

Maintenance software for the FMP, Loosely Coupled Network (LCN), and attached peripherals will reside on a disk in the Maintenance Control Unit. On-line maintenance software will be available for confidence testing and to support concurrent PM, as well as for emergency maintenance activities. Off-line maintenance software will be available to run margins and to support dedicated PM.

Maintenance software for the SPS will reside on magnetic tape and its organization will be similar to the FMP maintenance software.

LOGISTICS

Spare parts for the FMP and other system critical equipment will be stocked on site and in Minneapolis. Other high failure rate parts may also be stocked on site. Storage space for these parts and their cabinets is estimated to require 40 square feet.

System non-critical equipment parts will be stocked at one or more of the following locations: local service center, regional warehouse, Minneapolis distribution center. Distribution and quantities will be determined by part density and usage.

Emergency parts are generally available from the local service center in less than two hours and in less than 4 hours from the regional warehouse. Emergency orders for parts at the Minneapolis distribution center are filled within four hours, however, actual response is determined by airline schedules.

To minimize downtime and achieve the availability goals, the replaceable part will be the pluggable subassembly. To reduce costs, some of these assemblies may be repaired on site or at a local service center. The rest will be returned to a central repair facility in Minneapolis.

TECHNICAL SUPPORT

Technical support for the FMP will be supplied by the design and manufacturing division. The primary method of support will be through remote technical assistance (RTA). The RTA capability will be implemented through the use of data communication technologies for both on-line and off-line maintenance. Through RTA the supporting engineer will be able to manipulate diagnostics from a remote console. The remote console will be connected via phone lines to the NASF Maintenance Control Unit. All of the maintenance capabilities of the MCU will then be under control of the remote console. In this way problems requiring technical assistance can be analyzed directly by the supporting engineer.

Technical support for the SPS will be supplied in a similar manner. A remote console will be connected via phone lines to a multiplexer in the CYBER-175. Through this remote link assistance can be provided to the local customer engineer. Backup support will also be supplied by regional and headquarters support groups.

MAINTENANCE, NON-CDC EQUIPMENT

Maintenance of non-CDC equipment can also be supplied in most cases. This maintenance is under control of each individual region within Engineering Services. Maintenance for each equipment is subject to local availability.

DIVISION 8

MAINTENANCE SOFTWARE ALTERNATIVES FOR THE 1980s

DIVISION 8

MAINTENANCE SOFTWARE ALTERNATIVES FOR THE 1980s

1.0 INTRODUCTION

Reliability, Availability, Maintainability (RAM) needs for computer systems in the 1980s will focus on a reduction in the number of system interruptions as compared with today's systems and an overall lessening of impact on users when interruptions do occur. A requirement of one interruption per month is frequently stated for systems of the 80s as compared to several interruptions per day for current systems.

It can certainly be debated that it is economically feasible to reach a maximum interruption rate of one per month for medium size computer systems; however, super computer systems, designed for performance, present many more challenging problems. The key element is what the system does to modify/ease the effect on the user when an interruption arises. Suggestions are made to increase hardware/software self-monitoring, expand automatic (no manual intervention) system re-initialization and/or reconfiguration, enhance checkpoint/restart, and decrease dedicated system time needs for hardware and software maintenance.

With a decided trend toward more system time being available to the user and less system time being available for maintenance, it will be necessary to improve reliability through the techniques of fault-tolerant design, such as redundancy and self-diagnosis. These fault-tolerant techniques appear to be the key to the achievement of total on-line hardware and software maintainability so that a system would not need to be down for maintenance of any kind.

In short, computer system maintenance in the 1980s will need to be performed without significantly disrupting the system activity. It may be acceptable for the system and its users if response times degrade, but total unavailability of the system will not be acceptable.

Before discussing future alternatives, existing features of CDC super computer maintenance software will be addressed. The basic architecture includes a super computer, a maintenance station (control unit) as a focal point for all system maintenance, and a loosely coupled network (LCN) for I/O containing Programmable Device Controllers (PDCs).

2.0 EXISTING MAINTENANCE SOFTWARE OVERVIEW

2.1 General Features

The current maintenance software system supports off-line and on-line hardware maintenance. The major components of the maintenance software system are:

- Maintenance Control Unit
- CPU (Central Processing Unit) off-line diagnostics
- CPU on-line diagnostics
- PDC diagnostics
- Fault isolation
- Error logging and recovery

2.1.1 Maintenance Control Unit

The MCU (Maintenance Control Unit) which is built around a CYBER 18-20 Computer, provides a common user control point for all system maintenance activities. The MCU supports the following on-line and off-line maintenance activities through a local and/or remote terminal.

On-line	Off-line	Activity
X	X	Remote Maintenance
X	X	Display of MCU, PDC, and CPU Memory
X	X	Entry of MCU, PDC, and CPU Memory
X	X	Monitoring and Control of MCU and CPU Maintenance Lines
X	X	Logging of Memory SECDED Errors
X		Logging of Operating System Detected Errors
X	X	PDC Autoload
X		System Initialization
X		System Recovery
	X	Loading, Displaying, and Modifying CPU Microcodes
	X	Control of Off-line CPU Diagnostics
	X	Down-loading of Off-line PDC Diagnostics

2.1.2 CPU Off-line Diagnostics

The CPU off-line diagnostic system supports manufacturing checkout and field maintenance of the CPU. A structured set of diagnostics are available to provide error detection and analysis. Most CPU diagnostics have a built-in test mode to assist with remedial maintenance.

The object code of each diagnostic resides on MCU mass storage. The MCU loads each diagnostic into the CPU, MCU, or PDC memory as needed. Because the CPU diagnostics require a dedicated CPU, only one diagnostic can run at one time.

CPU off-line diagnostics consist of two types: fault detection diagnostics and utility diagnostics. The former test instructions and CPU resources, and form a testing hierarchy of the CPU. The latter support maintenance activities other than fault detection, such as test point simulators and error file analyzers.

2.1.3 CPU On-line Diagnostics

The CPU on-line diagostic system supports confidence testing of the CPU without shutting down the system. These diagnostics run as normal jobs. On-line diagnostics consist of all CPU based off-line diagnostics which do not execute monitor mode instructions. Also available for on-line use are test point simulators and error file analyzers.

2.1.4 PDC Diagnostics

The PDC off-line diagnostics test PDC components while the PDC controlware is inactive. The user can load these diagnostics from a portable maintenance console or can down-load them from the MCU to a PDC. Control of the PDC off-line diagnostics can come from the MCU or a portable maintenance console. PDC on-line diagnostics are controlled through the operating system. PDC functions are disabled while on-line testing occurs.

2.1.5 Fault Isolation

Isolation provides a definite means to reduce checkout time and MTTR. Fault detection is required while fault isolation is optional and depends on cost effectiveness. There are three major categories of isolation applied to diagnostics: physical isolation to the failing component, function isolation to the failing group of components, and unit isolation generally to the failing board.

Fault isolation to the failing memory chip for single solid faults is provided for all memory resources of the CPU. This includes central memory, register file, instruction stack and microcode memories. No isolation is provided for LSI arrays. Unit isolation is provided for critical units while functional isolation has rarely been used. Some unit isolation is provided in the LCN. The structure of the FMP is such as to provide very good unit isolation.

2.1.6 Error Logging and Recovery

The MCU serves as a local focal point for logging of all system errors. Separate files exist on the MCU for on-line and off-line errors. Time/date data is included in the on-line error file. Central memory SECDED errors are sent by hardware to the MCU for logging. Microcode memory parity errors are detected at the MCU. All errors other than memory errors are passed by the operating system to the MCU for logging.

Central memory can be degraded and reconfigured, through manual intervention from the MCU, when a solid fatal error occurs. A failing Vector Unit can be idled and replaced by a spare through MCU action. Any other fatal system error must be analyzed and corrected before system activity resumes. There are no provisions for reconfiguration of FMP hardware other than memory and the Vector Units.

2.2 Summary

In the 1970s the primary emphasis on maintenance software has been toward development of fault detection tests. Increased complexity of computer systems has led to development of the maintenance station, which is a local focal point for all system maintenance activities. With the development of complex LSI circuitry in mainframes and memory, fault isolation has begun to appear.

Maintenance software in general, and fault detection, fault isolation, error retry, and system recovery in particular, have been given low priority in design of system hardware and software.

In the 1980s hardware, operating systems, and maintenance software must be treated equally if the number of system interruptions is to be reduced. RAM requirements must enter into system design at the earliest stage, sharing the spotlight with cost and performance.

3.0 ALTERNATIVES FOR THE 1980s

To meet the challenge and requirements of improved RAM for super computers of the 1980s new alternatives must be considered. These alternatives must involve the entire system architecture, hardware, operating system, and maintenance software.

Ways must be found to minimize the mean time to detect and diagnose a system malfunction, the mean time to repair a detected malfunction, and the mean time to restart a system. Of high importance will be the ability to perform maintenance on, and repair of, a portion of the system without shutting down the entire system.

3.1 System Recommendations

The various levels of software and hardware must be designed with the following system goals in mind to minimize rerun time when system failures occur.

- Minimize the components required to continue processing -- i.e., minimize system critical hardware and software.
- Maximize system flexibility. Provide as much redundancy and alternate routes to accomplish the same functions as possible.
- Minimize the restart/recovery operations. Where possible, utilize the flexibility of a fall-back position by disconnecting system non-critical items and continue processing.

Actions taken for system error recovery are shown in Figure 1. Physical recovery is completely hardware dependent. All other actions involve use of on-line or off-line maintenance software, and interaction from a maintenance station to recover from an error. In cases where the system is reconfigured or degraded, concurrent maintenance software will support system repair.

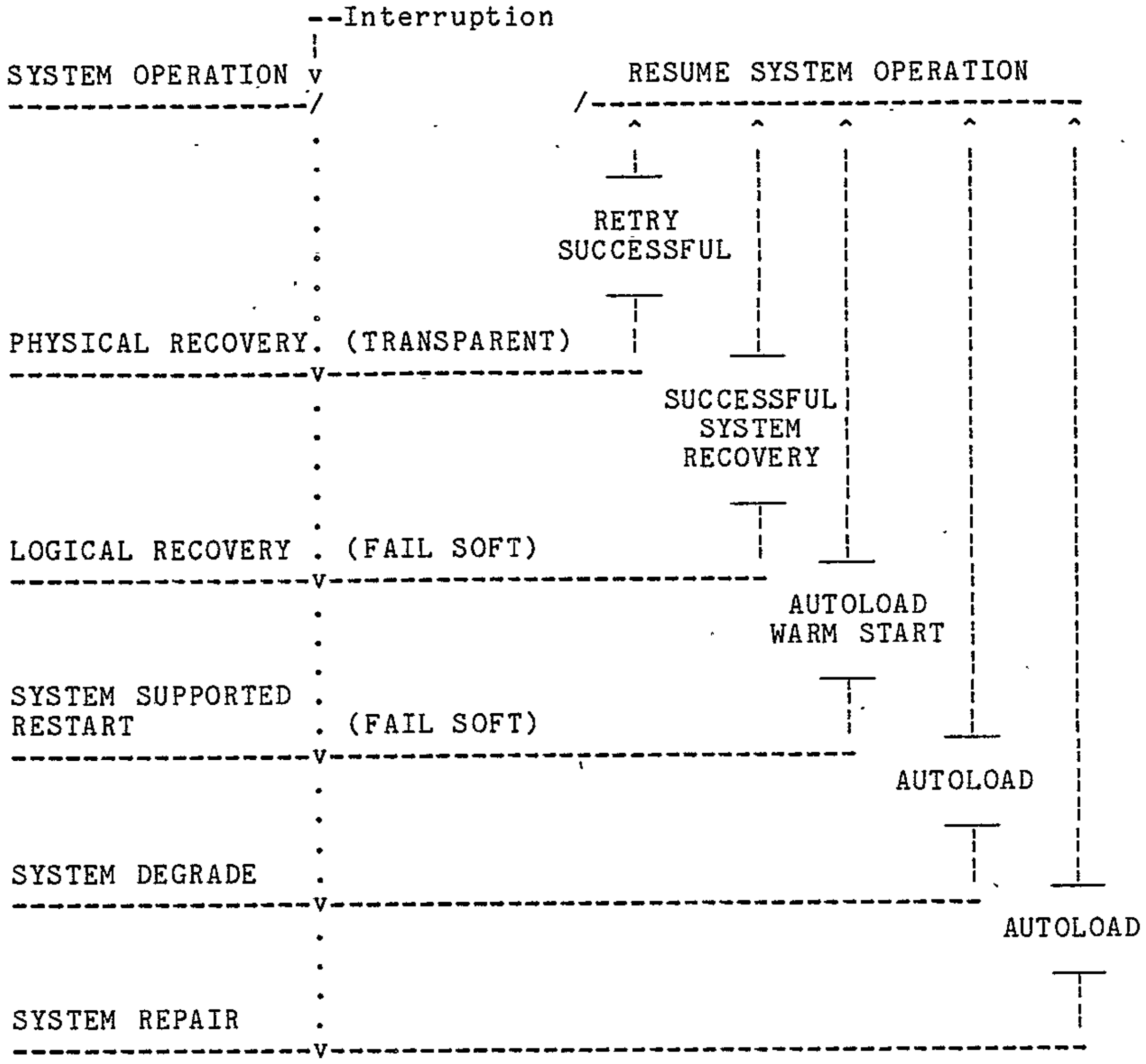


Figure 1. System Error Recovery

3.2 Hardware Recommendations

In the 1980s computer systems will have incorporated into their designs many of the hardware techniques mentioned below. At a minimum, 10% of the system hardware will be devoted to improved RAM.

- Semiconductor memories will continue to make use of single error correction, double error detection circuitry. Check bits will be carried along with data on all major trunks. Errors will be collected at a maintenance station such that preventive maintenance and degradation or reconfiguration can occur.
- Develop hardware maintenance features for high-level self-checking of semiconductor memories under control of the MCU. Memory testing would be independent of CPU instruction testing. Memory errors would be found faster under hardware control.
- Microcode control logic will be designed such that it is a useful maintenance tool. Microcode diagnostics will use added hardware features to assist maintenance personnel in isolating failures to the replaceable circuit level as quickly as possible.
- Major control paths may be encoded in error detecting codes to provide continuous fault diagnosis while executing programs. Error codes would aid development of fault isolation techniques.
- Transient faults may be identified by error detecting codes and their effects corrected by rollback. Permanent faults may be corrected by replacement of faulty units. Replacement may be automatic or under operator control from the maintenance console.
- Fault tolerant hardware involving the use of redundancy, with replication of individual circuits or subsystems, will appear. Fault tolerant hardware will mask the occurrence of random errors as they occur and provide error-free operation for large periods of time.
- Registers and latches in the CPU could be scanned by the maintenance station such that fault isolation to the chip is possible.
- Hardware maintenance features will appear in logic circuits, as more gates are placed inside circuits. The wafer probe and vendor testing problem will accelerate internal circuit testing needs.

3.3 Future Maintenance Software Development

The demands of the computer industry are for improved mean time between interruptions and a reduction in lost time. Maintenance software must be treated as a system working in conjunction with hardware and the operating system to meet these demands.

To achieve these goals maintenance software development emphasis will be placed in the following areas:

- Fault isolation
- Operational summation
- Loosely coupled network I/O
- Application of gate simulation data base
- On-line
- Error logging
- Recovery
- Concurrent maintenance

3.3.1 Fault Isolation

Isolation must be specified as part of the hardware design requirements. Error detection codes, microcode hardware features, multiplexing of registers to the MCU must be considered if fault isolation diagnostics to the function and circuit are to be developed.

3.3.2 Operational Summation

Without changes in hardware design philosophy, operational summation can go far to improve MTTR. With a status of failing and operational CPU instructions, CPU resources, and I/O resources, maintenance personnel can draw upon their knowledge of the system to localize the fault. Maintenance action would be based on operational summation or decision logic tables.

3.3.3 Loosely Coupled Network I/O

The MCU must be capable of testing selected PDCs and associated I/O devices concurrently with CPU testing. Communications within the I/O network should also be testable from the MCU. In view of the sophisticated protocol between PDCs, special maintenance software considerations must be made.

3.3.4 Application of Gate Simulation Data Base

Hardware modeling and simulation is a requirement for super computer design; therefore, a gate model of the CPU is available for maintenance software applications. The following lists possible applications:

- Routing/placement information. . . the user could query the model from the MCU display.
- Simple simulations could be performed by the MCU to generate test point states based on input operands.
- LSI fault isolation. . . compare the hardware to the model.

3.3.5 On-line

The following improvements can be made to existing fault detection diagnostics:

- Reduce the risk of diagnostics failing in test condition setup, or in result verification by developing standard coding methods.
- Expand on diagnostic data base concepts where a test condition data base can be controlled from either the CPU or MCU. This method of testing should ultimately replace the existing computer command tests.
- Develop a high-level test of the basic housekeeping instructions used to control CPU based diagnostics, including mixed instruction testing. This level of testing should be controlled from the MCU.
- For the LCN, provide common maintenance software and procedures for the entire network, including super computer, front-end systems and peripherals.
- Develop common maintenance procedures for all field sites, particularly the procedure to follow when a fatal system error occurs.

3.3.6 Error Logging

Error messages should be organized as 4 basic types: (1) MCU detected errors (CPU hardware fault detection), (2) CPU errors detected by the operating system, (3) LCN and I/O device errors, and (4) logging of software system errors. Common error files should support error logging for multi-CPU systems.

3.3.7 Recovery

More emphasis should be placed on automatic recovery and error logging for deferred maintenance. Since most system errors are transient or intermittent, recovery with degradation is a viable alternative to increase system availability. System software must be enhanced to promote automatic restart. Only fatal errors in the operating system should force emergency maintenance.

Automatic degradation and reconfiguration should be provided for central memory, I/O, and hardware pipelines. The CPU could be temporarily stopped by the MCU until degradation/reconfiguration takes place. The following degradation alternatives are offered:

- Page flawing for central memory
- Central memory degradation by physical sections
- Dynamic testing/flawing of 819 tracks, particularly on initial file creation
- Provide a spare pipeline which can be switched to an active pipe by the MCU
- Spare PDCs between the trunk and central memory

3.3.8 Concurrent Maintenance

Before concurrent maintenance of CPU units (memory, pipelines) can become a reality, hardware must be designed such that maintenance actions can occur on degraded units without affecting operating units.

Concurrent maintenance of degraded PDCs interfaced with the CPU from the MCU seems realistic. Concurrent maintenance of critical I/O devices would have the greatest impact but unless the system can tolerate downed devices, this alternative is unlikely.

4.0 IMPLEMENTATION GOALS AND STRATEGY

The following goals and strategies will be pursued to improve super computer RAM in the 1980s.

4.1 Goals

- For system availability, improvements to MTBI are deemed more effective than improvements to MTTR; therefore, super computers will stress improvements to MTBI.
- MTTR of less than 0.5 hours for all levels of memory.
- MTTR of less than 1.0 hour for CPU (excluding memory) and I/O.
- Achieve a system MTBI of greater than 100 hours.
- Fault isolation.

Hardware	% of Single Solid Faults Isolatable	Isolation Level
Memories	100%	Board & Circuit
PDC	90%	Board
LSI Logic	50%	16 or less Circuits
LSI Logic	90%	Functional Unit

- Fault detection, on-line.
  - -- 99% of solid hardware faults are detectable by the operating system, maintenance software, and hardware self-checking features.
  - -- 50% of transient faults are detectable by the operating system and hardware self-checking features.
- Fault detection, off-line.
  - -- 99.5% of solid faults are detectable by maintenance software and hardware self-checking features.
  - -- 85% of intermittent faults are detectable by maintenance software and hardware self-checking features.

4.2 Strategy

The following strategy will be followed to satisfy the previous goals. New development projects will emphasize, but not be limited to these areas, and efforts must be made to improve existing products in these areas.

4.2.1 Hardware Strategy

- Develop hardware maintenance features for fast, high-level self-testing of large semiconductor memories.
- Design microcode control such that it is a useful maintenance tool placing emphasis on fault isolation.
- Develop hardware to multiplex CPU registers to the MCU.
- Develop fault detection techniques in CPU and LCN hardware modules and all data paths.
- Configure CPU and LCN hardware modules such that it is possible to remove faulty modules from operation while minimizing impact to the operational system.

4.2.2 Operating System Strategy

- Improve error detection and error logging techniques.
- Improve system error recovery techniques, stressing fail-soft recovery with system reconfiguration.

4.2.3 Maintenance Software Strategy

- Develop fault isolation diagnostics, using added hardware features.
- Improve on-line maintenance software.
- Develop concurrent maintenance diagnostics.
- Improve remote maintenance capabilities.
- Reduce time required to verify system operation.

4.3 Attainability

A high priority effort in the design of new hardware systems (including the FMP) is the provision for integral maintenance "hooks" throughout such machines. Cost and design resource factors have taken this feature into account. The degree to which these facilities will be exploited will, however, depend on the resources committed to sophisticated maintenance strategies by the various software developers. The inclusion of features in the various levels of software (from device driver all the way up to applications program instrumentation) is feasible and within the range of known software techniques. The probability that the NASF will possess the maximum of these capabilities rests solely on management commitment by NASA and its contractors to assigning these facilities a high priority in the implementation program.

DIVISION 9

INSTALLATION ORGANIZATION/OPERATION

DIVISION 9

INSTALLATION ORGANIZATION/OPERATION

The information in this division of the report contains a recommended data center organization for NASA planners to aid in the formulation of an accurate life-cycle model for the NASF installation. Data presented in this division was gathered from the personnel of Control Data's STAR-100 Data Center in Arden Hills, Minnesota, a data center possessing similar characteristics to the proposed NASF facility.

In setting up an effective new data center a key point is the pursuit of extensive research on the users, types and frequency of jobs, and projected increases in work. Once the user base has been established, as well as whether remote stations will be employed, data center personnel can more accurately plan the installation.

The following information, which is provided to assist in the planning of NASF Operations, is broken down into manpower, supplies, services, and an overall scenario that will attempt to show typical considerations in data center management. It is hoped that this information will enable NASA planners to set up a NASF Data Center with an eye for efficiency, reliability, and stability.

MANPOWER REQUIREMENTS

Based on information obtained from Control Data large system data center activities, it is recommended that NASA adopt an organization similar to the one shown in figure 1. This suggested organization has four managers controlling the operations, system support, techniques support, and administrative and technical support -- all reporting to the overall NASF center manager.

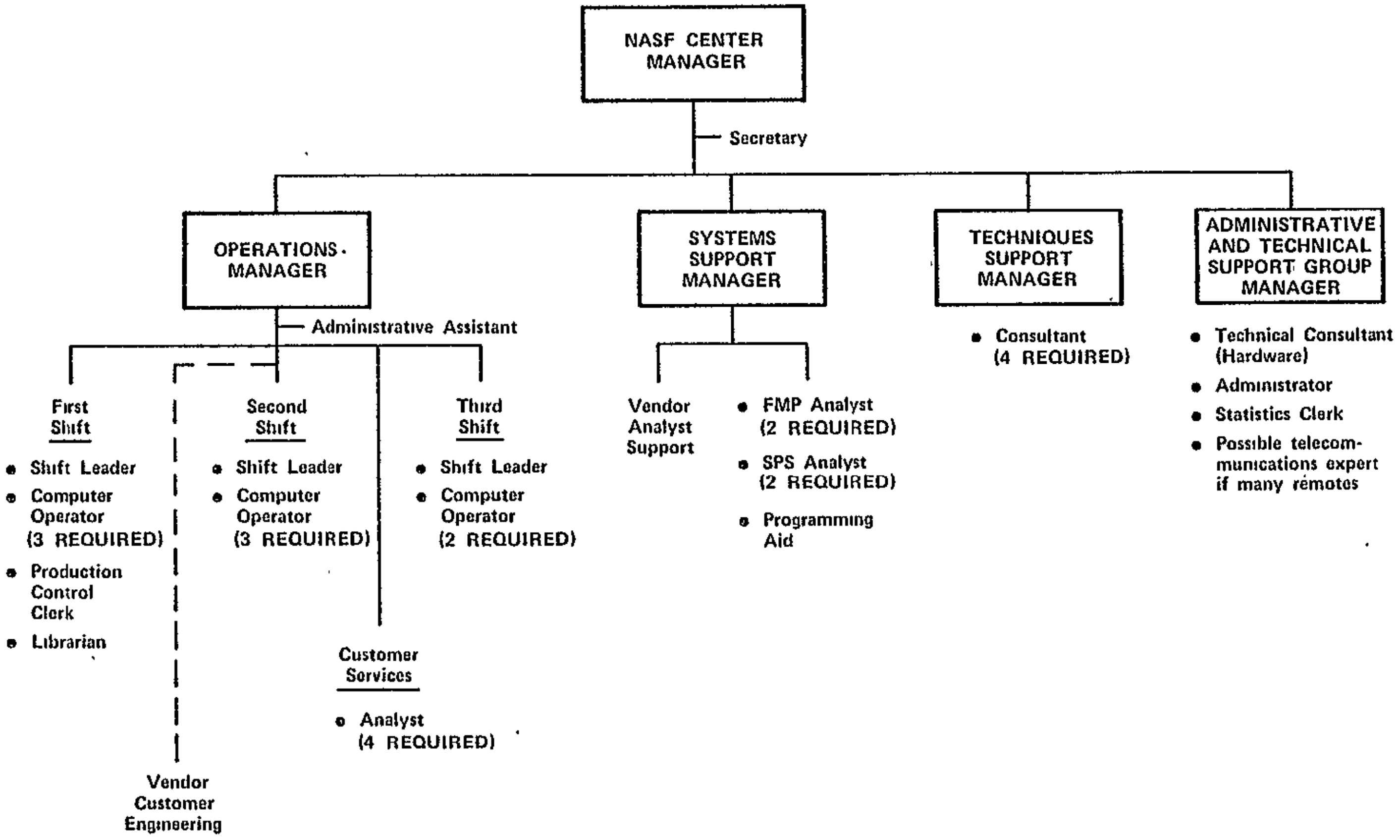


Figure 1. Suggested NASF Operations Organization

The Operations Manager

The operations manager is responsible for ensuring the day-to-day flow of customer jobs through the data center in a timely manner. The main duties of the operations manager are:

- to set up operating procedures for efficient data center operation;
- to schedule addition of local modifications or Field Change Orders with customer engineering staff;
- customer or user support scheduling;
- customer service dealing with customer problems related to operations;
- to monitor daily operations attempting to foresee both hardware and software problems;
- to deal with vendor customer engineers.

Within the operations organization, the manager must provide leadership to the following subordinates.

Shift Leader -- The shift leaders (3 in the organization chart) are the most experienced computer operators on the shift. They are usually higher pay-grade levels than the other operators on the shift and could act as assistant operations manager because of the amount of experience they have.

A good shift leader's experience would allow him/her to be familiar with all aspects of the various jobs run in the center and he/she should have extensive training on the center's equipment.

Computer Operator -- The operators (eight in the organization chart) should be trained on the NASF and be able to complete the tasks related to efficient customer job throughput.

Process Control Clerk -- The process control clerk is responsible for keeping track of the input and output of customers' jobs within the data center. Often times the process control clerk, in addition to scheduling duties, will function as the requisitioner of supplies needed within the data center for its day-to-day operations, i.e., tab cards, line printer paper, office supplies, etc.

Librarian -- The librarian for the data center sets up and maintains a filing system for magnetic tapes, disk files, and card decks that are often used, thus freeing user time normally spent in this function.

Customer Service Analyst -- Two to four analysts should be available to troubleshoot problems that customers may have in getting their jobs successfully completed. The number of analysts employed at the data center will be determined by whether or not several remote sites are used and whether the major part of the customer input will be at the data center itself.

The customer service analyst is the first person in the data center the customer should contact if problems arise. These analysts should be of several grade levels so that more complex problems can be addressed by more senior, experienced analysts.

The Systems Support Manager

The systems support manager oversees high-level analyst support of the system and, along with his systems analysts, gets the system operational and solves problems the customer may have that go beyond those encountered in operation.

Responsibilities of the systems support manager include:

- building and maintaining the operating system, or systems, used in both the FMP and SPS sections of the system;
- ensuring that Program Trouble Reports (PTRs) have been properly resolved;
- isolating problems in the system so that Customer Engineering or an analyst can correct it;
- providing vendor analyst support if needed.

In order to effectively deal with the analysts reporting to him/her, the systems support manager should be software knowledgeable.

The systems support manager must supervise the activities of:

FMP Analyst -- These analysts should be familiar with the total system but should be most familiar with the FMP. They would deal with customer job problems of an intermediate nature that have been localized in the FMP. They would also build and maintain the FMP operating system and be familiar with the SPS operating system as backup.

SPS Analyst -- These analysts should be familiar with the total system but should be most familiar with the SPS. They would deal with customer job problems of an intermediate nature that have been localized in the SPS. They would also build and maintain the SPS operating system and be familiar with the FMP operating system as backup.

Programming Aide -- The programming aide assists the analysts in the successful completion of their jobs.

Techniques Support Manager

The techniques support manager should be a high-level analyst fully capable of solving the most complex problems the customer could have. Likewise, his crew of consultants should be top-notch people of the highest level.

The techniques group is responsible for:

- working with customers on setting up their basic software;
- acting in a quality assurance function for customer's software to make it more efficient if necessary;
- providing education so that customers may make efficient use of the system;
- setting up benchmarks for vendors;
- generating an algorithm base.

Consultants -- Reporting to the techniques manager are four consultants who should be experts in at least one area of responsibility of the techniques group, e.g., physics, aerodynamics, structures, meteorology. Generally, the techniques consultants provide direct support to the user groups.

Administrative, Technical Support Manager

The administrative and technical support manager is basically the business manager for the data center. Functions that fall into this area of responsibility include:

- initial configuration and revision of existing configurations;
- reports and statistics needed for various management reports;
- billing and other transactions with accounting personnel;
- technical support -- primarily for evaluation and purchase of new equipment.

Reporting to this manager are the following personnel:

Technical Consultant (Hardware) -- This consultant must be familiar with all facets of hardware so as to be the hardware resource person for data center management. The ability to configure and reconfigure a wide variety of hardware is important, especially at the time of the initial data center setup. If it is expected that the data center will be static as far as new equipment is concerned, this function may be eliminated. However, if several remote locations are to be tied into the data center, this technical expert perhaps should at least be familiar with telecommunications, which will become increasingly more important as the amount and sophistication of the remote equipment grows. In addition, this consultant should be able to develop both short- and long-range budgets, and deal with the technical aspects of moves in the data center.

Administrator -- The administrator would probably have direct responsibility for ensuring that reports, billing, and other tasks get completed. This person is often the interface between the data center and other supporting departments such as plant maintenance or accounting.

Statistics Clerk -- This person would generate reports based on data furnished by others within the data center.

The job functions previously discussed are not meant to be only as described. The exact functions should be flexible enough so that they overlap. All data center personnel should complement each other to assure a smoothly functioning organization.

The cost of personnel to adequately staff a data center of the size proposed to NASA will be dependent on NASA's pay structures. A recent study of the data processing industry has shown that personnel costs are as high as 50 percent of a data center operating budget.

Increases to personnel wages are constrained by U.S. Government wage guidelines, hardware costs are decreasing, and increasing shortages of good, high-quality people will make accurate forecasting difficult or impossible for the near future.

DATA CENTER SUPPLIES

Any discussion of supply usage will, of course, depend on the way the data center is used and placed among its users. If it is to be in or near the greatest number of users, more supplies such as printer paper, punched cards, and magnetic tapes will be used. If the center will be separated from the bulk of users, more remote stations will be utilized thus reducing supply expenditures at the center itself.

In general, Control Data recommends stocking an initial amount of magnetic tapes. After the data center is functioning efficiently, CDC's data center staff estimates that a monthly tape replacement of about 10-15 tapes would be likely.

Control Data's Mass Storage Sytem (MSS) is recommended for the NASF installation. Use of the MSS results in:

- decreased use of disk space,
- decreased operator time or possibly requirements,
- decreased tape expenditure.

The MSS concept is relatively new and CDC has not yet compiled enough information on its data center utilization to accurately predict cartridge replacement frequency; as with magnetic tape, an initial supply should be stocked.

The other big supplies expenditures would be in line printer paper, printer ribbons, and punch cards. The following chart lists NASF installation projected monthly usage of supplies for both a center without many remote stations and a typical remote station that orders its own stock of supplies.

Item	Data Center Use/Month	Remote Station Use/Month
Printer Ribbons	25	7
Printer Paper	225 boxes	50 boxes
Punch Cards	120 boxes (5 boxes per case)	30 boxes (5 boxes per case)
Magnetic Tapes	10 to 15	N/A
Mass Storage Cartridge	Data not available	Data not available

The other major expenditure for a functioning data center would be in capitol equipment replacement and purchase. This will include the purchases of new updated equipment to make the data center more efficient, as well as replacement of the often-used old equipment.

SERVICES

Other projected needs of the users that may be provided by the data center include:

- user communications newsletter
- keypunching, interpreting, etc.
- monitoring of remote sites
- special delivery services to include:
  - personal delivery to the airport
  - packaging for shipment
  - interfacing with shippers
- listing, binding, collating, and bursting
- microfilm/replication services

DATA CENTER SCENARIO

The planning of the data center staff must include a mix of position levels, as indicated previously, to allow growth from within the organization. Also, a good mix of low-level to high-level technical content positions give room for personal growth.

One can expect impact on the daily functioning of the data center due to vacation, sickness, tardiness, etc. A backup strategy can be accomplished in one of two ways.

1) Staff "lean-and-thin" which will require overtime by others during times of personnel shortages.

2) Over-staff so that absence of a couple people will not noticeably impact schedules.

In place of distinct eight-hour shifts, it is possible to define a variety of shifts, allowing the data center to attract computer operators who prefer to work unusual hours. This method also provides better transition of data center operators throughout the day. Refer to figure 2 for an example.

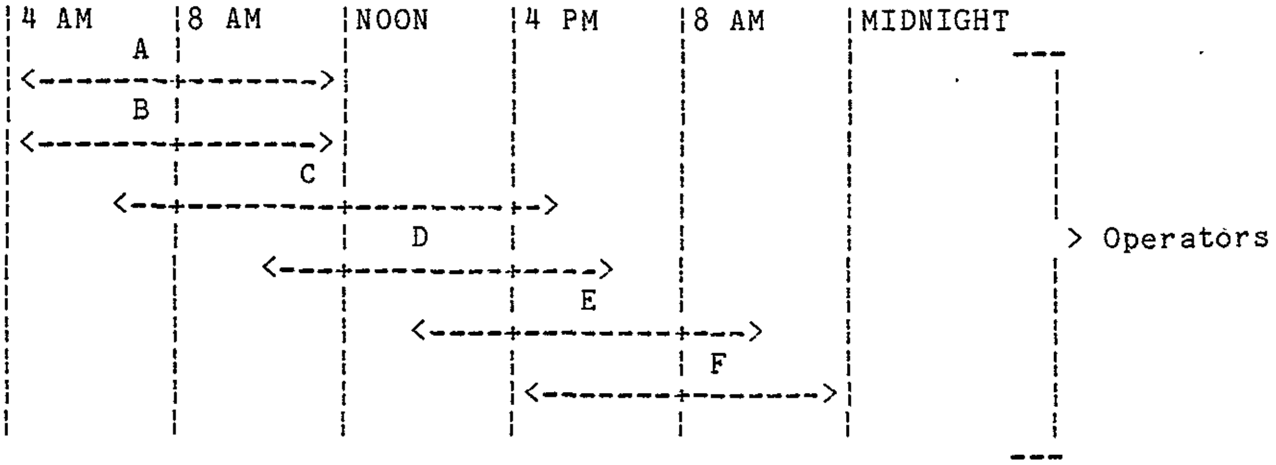


Figure 2. Data Center Operator's Schedules

By use of overlapping shifts, operators are more flexible and the data center can be more efficiently run with less difficulty adjusting for absences. Also, the overlapping of computer operator hours allows for more communications between the operators than does a static eight-hour shift.

Computer operators at such a center should be considered as professional or semiprofessional -- not as blue collar automation. Thus the level of system responsibility and understanding is higher.

Reliability and stability data must be provided via the operating system software. Since this will be an ongoing concern, this should be addressed immediately upon startup of the installation.

The design of the NASF center with regard to storage and office space will have to be considered in terms of NASA policies. CDC recommends an operator's desk be in the center and that one desk be supplied for every two operators on a shift. Care must be taken in designing adequate, well-lighted space for visitors to work on their jobs as well.

Controlled storage space, preferably within the range of the computer room air-conditioning, should also be provided. Supplies, especially paper products, need to be stored in a constant temperature, low humidity environment.

Other considerations for space allocations that CDC believes are important are:

- Limited- or controlled-access storage for cards, tapes, etc.
- Room for several users to lay out their work and use terminals.
- Adequate counter space for several users to discuss jobs with center personnel.
- Keypunches for quick "fixes" of user jobs.
- Offices for the support or customer services analysts with offices that open into the center itself.
- Local terminal capabilities so that users may be able to make minor modifications to their work.

More definitive information on floor layouts and environmental considerations will be found in Division 10.

SUMMATION

In addition to the data already presented here, the following points should also be considered by NASA.

1) A hardware calendar clock is recommended by CDC. This is an aid in reducing purged files caused by an operator mistakenly typing in the wrong date. Several large Government Data Centers have this clock installed in their systems.

2) Because of the complexity of the NASF system, short-term, part-time operators are discouraged. The return on the training effort is most often poor. Part-time operators that are employed should be long term ones for this reason. Part-time operators do, however, lend a good deal of flexibility to manpower scheduling in the data center.

These points, plus the other information in this division, should enable NASA planners to begin to visualize the structure of the NASF facility. As the NASF plan further materializes, more specific recommendations and estimates can be made by Control Data.

DIVISION 10

NASF PHYSICAL REQUIREMENTS UPDATE

DIVISION 10

NASF PHYSICAL REQUIREMENTS UPDATE

The report for the first study of this project (ref. 1) presented size, power, and cooling requirements for the NASF. This report provides an update of those requirements; it consists primarily of two tables. Table 1 shows a detailed listing of all the elements that make up the SPS, the disk station, and the MCU, including all the anticipated peripheral equipment. All the equipment shown are products today. Table 2 includes all the separate parts that make up the FMP computer system. The motor-generator sets and the condensing units exist today but the numbers for the parts of the FMP -- computer bay, Main Memory, Intermediate Memory, and Backing Store -- are today's best estimates since these items have yet to be built.

All the numbers in the tables are subject to change as design proceeds and details of the FMP, as well as the system, become more clearly defined.

Table 1. Standard Product Physical Requirements

```
CONTROL DATA CORPORATION                                                                         COMPUTER FACILITY PLANNING AND CONSTRUCTION
DOCUMENT NO.                                                                                                                 PAGE    1
RUN DATE  03/22/79                                           MACHINE UNIT SPECIFICATION
CUSTOMER: NASA AMES                        TOTAL SYSTEMS

                                                                                                   UNIT CIRCUIT BREAKER REQUIREMENTS
                                        --------PHYSICAL PROPERTIES-------  UNIT                  T (02)                     C             C
                                        UNIT   UNIT   UNIT    UNIT   UNIT    HEAT                  P 400HZ    50/60HZ        O   50/60HZ   O
 CABINET NAME                       QTY WIDTH  DEPTH  AREA   HEIGHT WEIGHT DISSIP  ---UNIT KVA---  W 208V,3P  208V,3P       N   115V,1P   N
  MODULE/MODULES       PRODUCT          (IN.)  (IN.) (SQ FT) (IN.)  (LBS) (BTU/HR) 400HZ  50/60HZ  R AMPERES  AMPERES       N   AMPERES   N
 CENT COMPUTER BAY 1
                       175-116        2  93.70  35.00   45.55 79.80  4500   3660     10.8    -     R  50
 CENT COMPUTER BAY 2
                       175-116        2 101.90  35.00   49.53 79.80  3600   5860     17.3    -        70
 CONSOLE
                       175-116        2  32.50  47.00   21.22 48.50   390   3030      0.8   0.3       15                             15(08)  (02)
 CONDENSING UNIT
                       175-116        2  72.00  26.00   26.00 48.00  1480 174000(13)   -   14.4                  70(0[illegible]) (02) :
 MAG TAPE CONTROL
                       7021-32        1  29.30  30.00    6.10 60.00   250   2750      0.6   0.2       15                             15(08)  (02)
 REMOTE PROCESSOR CAB
                       7021-32        1  29.30  30.00    6.10 60.00   350   2750      0.6   0.2       15                             15(08)  (02)
 MAG TAPE TRANSPORT
                       677-4          2  30.50  30.00   12.71 63.50   900   7920       -    2.9                  15(0[illegible]) (03) :
 MAG TAPE TRANSPORT
                       679-7          2  30.50  30.00   12.71 63.50   900   7920       -    2.9                  15(0[illegible]) (03) :
EXTENDED CORE STORAGE
 PERIPHERAL CONTROLLER   B CAB
                       7030-104C      1  42.00  20.50    5.98 56.90   625   3800      0.9   0.4       15                             15      (02)
 STORAGE CABINET 1
                       7030-104C      1  70.00  40.80   19.83 72.50  1600  34400(12)   7.2   3.6       30        20       (02)
 DDP CONTROLLER
  PO ECS CONFIG.       7030-104C      1  29.00  25.00    5.03 66.00   500   1400      0.3   0.1       15                             15      (02)
 CHANNEL CONVERTER
   (3000)              10315-1        4    -      -       -     -      -      -         -     -    R
 CHANNEL CONVERTER
   (3000)              10315-2        4    -      -       -     -      -      -         -     -    R
 TRANSFER SW CONT
                       3270A          1  22.90  20.50    3.26 56.90   450   2000       -    0.6    R                                15      (02)
 TRANSFER SWITCH:
  IN 3270A/B           8271-2         2    -      -       -     -      -      -         -    0.6    R           PWR FR M 3270
 NETWORK PROCESSOR
                       2551-1         2  24.00  34.50   11.50 75.00   700   6480       -    1.9                                     20      (02)
 CARD PUNCH
  250 CPM              415            1  21.50  39.50    5.90 45.00   568   3000       -    1.1                                     15      (03)
 PERIPHERAL CONTROL
                       3446-2         1  42.00  20.50    5.98 56.90   650   2700      0.7   0.1    R  15                             15      (02)
 CARD READER/CONTROLLER
  1200 CPM             405            2  57.00  33.00   26.13 46.00  1200   9000       -    3.4    R              15       (03)
 CARD READER CONT
  IN 405               3447-2         2    -      -       -     -      -      -         -     -    R           PWR FR M 405 CR
 LINE PRINTER/CONT
  2000 LPM             580-200        4  62.00  31.50   54.25 51.50  1500  15000       -    5.2    R             30(0[illegible]) (02)
 MOTOR GENERATOR
   80 KVA        ,(0)165045-1KC       1  32.00  32.00    7.11 60.30  3875  47800(11)   -     -                IN/OUTPUT CONN TO MG CONT
```

Table 1. Standard Product Physical Requirements (Continued)

CABINET NAME MODULE/MODULES	PRODUCT	QTY	PHYSICAL PROPERTIES UNIT WIDTH (IN.)	UNIT DEPTH (IN.)	UNIT AREA (SQ FT)	UNIT HEIGHT (IN.)	UNIT WEIGHT (LBS)	UNIT HEAT DISSIP (BTU/HR)	UNIT KVA 400HZ	UNIT KVA 50/60HZ	UNIT CIRCUIT BREAKER REQUIREMENTS TPWR (02) 400HZ 208V,3P AMPERES	50/60HZ 208V,3P AMPERES	CONN	50/60HZ 115V,1P AMPERES	CONN
MG CONTROL CABINET	65045-1KC	1	56.00	23.00	8.94	74.00	1050	- (11)	-	-		SIZE FOR 125HP/93KW MOTOR			
TERMINATOR POWER (WALL MOUNT)	18182800	1	24.00	6.80	1.13	12.00	50	-	-	-	15				
DEW POINT RECORD (WALL MOUNT)	53370000	1	15.50	7.50	0.81	36.00	80	-	-	-				15(08)	(02)
TEMP MON/PWR CONT (WALL MOUNT) PO CY170	53368900	1	25.00	7.00	1.22	16.00	40	-	-	-		PRIMARY PWR GRP OR EM OFF			
EMERGENCY OFF (WALL MOUNT)	53369800	1	21.00	7.90	1.15	12.00	35	-	-	-		15(07)	(02)		
CENTRAL PROCESSOR	18-20	1	61.00	31.00	13.13	29.00	475	3820	-	1.1				15	(04)
MOS MAIN MEMORY	1882-32	2	-	-	-	-	-	-	-	-		IN CPU			
STORAGE DRIVE INTRF	1833-1	1	-	-	-	-	-	-	-	-		IN CPU			
CARD R/LINE PR CONT	1828-1	1	-	-	-	-	-	-	-	-		IN CPU			-
COMM LINE ADAPTER	1843-1	1	-	-	-	-	-	-	-	-		IN CPU			
MAG TAPE SUBSYS	1860-1	1	22.50	32.00	5.00	68.00	425	1640	-	0.6				15	(04)
MAG TAPE TRANSPORT	(09) 1860-92	1	19.00	18.00	2.38	24.00	225	1640	-	0.6					(02)
INSTALLATION KIT	1860-201	1	22.50	29.50	4.61	68.00	300	-	-	-				15(06)	(04)
MAG TAPE TRANSPORT	677-4	1	30.50	30.00	6.35	63.50	900	7920	-	2.9		15(07)	(03) :		
MAG TAPE TRANSPORT	679-7	1	30.50	30.00	6.35	63.50	900	7920	-	2.9		15(07)	(03) :		
CARD READER	1829-60	1	14,20	19.00	1.87	16.50	55	1310	-	0.5				15(08)	(04)
LINE PRINTER	1827-60	1	34.00	26.50	6.26	44.50	300	3275	-	1.2				15(06)	(04)
DISPLAY TERMINAL	1811-2	1	21.60	20.40	3.06	15.20	51	430	-	0.1				15	(04)
DISK STORAGE UNIT	819-21	16	27.00	45.00	135.00	45.00	800	5120	-	1.5		20	(03)		
MASS STORAGE CONT	7639-22	4	29.00	25.00	20.14	66.00	450	2840	0.5	0.7	15			15(08)	(02)
STORAGE MODULE DRIVE	1867-20	2	22.00	36.00	11.00	36.20	218	2355	-	0.8				15	(04)
STORAGE DRIVE CONT IN FIRST DRIVE	1833-3	1	17.70	24.00	2.95	12.00	91	1030(11)	-	0.3				15	(04)
NON-NUMERIC DATA IN DATA BASE	885-12	4	33.30	42.00	38.85	44.40	1080	-	1.6	2.2	15	15(07)	(10)	(08)	

ORIGINAL PAGE IS OF POOR QUALITY

Table 1. Standard Product Physical Requirements (Continued)

```
DOCUMENT NO.
                                                                                                                        PAGE   3

                                                                                                      UNIT CIRCUIT BREAKER REQUIREMENTS
                                  --------PHYSICAL PROPERTIES-------- UNIT                          T (02)                    C          C
CABINET NAME                      UNIT   UNIT   UNIT     UNIT   UNIT    HEAT                        P 400HZ    50/60HZ        O  50/60HZ O
 MODULE/MODULES    PRODUCT    QTY WIDTH  DEPTH  AREA     HEIGHT WEIGHT  DISSIP    ---UNIT KVA---    W 208V.3P  208V.3P        N  115V.1P N
                                  (IN.)  (IN.)  (SQ FT)  (IN.)  (LBS)  (BTU/HR)  400HZ  50/60HZ     R AMPERES  AMPERES        N  AMPERES N
FMD CONTROLLER
                   7155-1       2  29.00  25.00   10.07  66.00   350    3310      0.9    1.1          15                         15(08) (02)
MASS STORE ADAPT
                   7880-1       3  53.80  31.00   34.75  71.00   500   10000      -      3.0      A            30(0 ) (03)
MASS STORAGE COUPLER
                   7880-1       3  29.20  25.00   15.21  66.00   500    1435      0.3    0.1          15                         15(08) (02)
CARTRIDGE STORE UNIT
                   7881-1       8 128.00  21.00  149.33  76.80  2030    6000      -      1.7                       15
CARTRIDGE TRANSPORT
                   7882-1      16  26.80  31.00   92.31  71.00   700   10600      -      3.1                       15(08) (02)

                                                                                  OTAL  AREA(EQUIP ONLY) =      874.5 SQ FT
                                                                                 TOTAL  WEIGHT =    92.665 LBS
                                                                                 TOTAL  HEAT DISSIPATION =   233.910 BTU/HR
                                                                                 TOTAL  KVA(400HZ) =    79.2
                                                                                 TOTAL  KVA(50/60HZ) =   291.7

                                                                                  OTE - ABOVE TOTALS EXCLUDE ANY ADDITIONAL DATA
                                                                                 THAT MAY ACCOMPANY THIS SPECIFICATION (SEE BELOW)
```

Table 1. Standard Product Physical Requirements (Continued)

```
DOCUMENT NO.
NOTES:
 - ALL SPECIFICATIONS ARE ON A PER UNIT BASIS
 - WITH EXCEPTION OF UNIT AREA(REFLECTS TOTAL UNIT AREA)
(A) INTERNAL TERMINATOR POWER SUPPLY
(B) EXTERNAL TERMINATOR POWER SUPPLY
(01) MAXIMUM CAPACITY SHOWN. ACTUAL LOAD MAY BE LESS.
(02) TERMINAL STRIP.
(03) LOCKING CONNECTOR.
(04) STANDARD 60HZ PLUG.
(05) 380V-3PH. FOR 50HZ VERSION.
(06) 220V - 1PH. FOR 50 HZ VERSION.
(07) 380/415V - 3 PHASE + N FOR 50HZ OPERATION
(08) 220/240V - 1 PHASE + N FOR 50HZ OPERATION
(09) RACK MOUNTABLE EQUIPMENT
(10) CUSTOMER SUPPLIED CONNECTORS
(11) INDICATED EQUIPMENT IS NOT INCLUDED IN MUS SUMMARY TOTALS
     FOR AREA,HEAT AND POWER
```

```
(12) WATER COOLED-90 PERCENT OF HEAT REJECTED TO WATER.
                  10 PERCENT OF HEAT REJECTED TO ROOM.

                             *************************************
                             * INLET TEMP * FLOW RATE * HEAD LOSS*
                             * DEG F DEG C* GPM  L/MIN* PSI  KPA *
     WATER REQUIREMENTS      *  80     27  * 4.2  16.0 * 5.7   39 *
          PER BAY            *  70     21  * 3.0  11.5 * 2.6   18 *
                             *  60     16  * 2.0   8.0 * 1.5   10 *
                             *************************************
                            HEAD LOSS DROP FOR OPEN CONDENSOR ONLY.
                            MINIMUM OPERATING PRESSURE DIFFERENTIAL
                            AT UNIT WATER CONNECT IS 10 PSI/69 KPA.
                            MAXIMUM WATER PRESSURE 100 PSI/690 KPA.


(13) WATER COOLED-161000 BTU/HR (47170W) REJECTED TO WATER.
                  13000 BTU/HR (3809W) REJECTED TO ROOM.

                             *************************************
                             * INLET TEMP * FLOW RATE * HEAD LOSS*
                             * DEG F DEG C* GPM  L/MIN* PSI  KPA *
          10 TON             *  80     27  *12.0  46.0 * 9.5   65 *
      CONDENSING UNIT        *  70 .   21  * 9.3  35.3 * 6.3   43 *
                             *  60     16  * 7.6  28.8 * 4.3   30 *
                             *  50     10  * 6.2  23.5 * 3.4   23 *
                             *************************************
                            HEAD LOSS DROP FOR OPEN CONDENSOR ONLY.
                            MINIMUM OPERATING PRESSURE DIFFERENTIAL
                            AT UNIT WATER CONNECT IS 15 PSI/103KPA.
                            MAXIMUM WATER PRESSURE 100 PSI/690 KPA.
```

```
ADDITIONAL DATA-
```

Table 2. FMP Physical Requirements

		Unit Dimensions inches (cm)					
Qty	Cabinet Name	Width	Depth	Height	Unit kVA 60 Hz	Unit Dissipated Heat BTU/hr (kcal/hr)	Notes
1	FMP Computer Bay	146 (370)	102 (259)	76 (193)		331K (83K)	(1)
1	FMP Main Memory	111 (282)	63 (160)	76 (193)		576K (145K)	(1)
1	FMP Intermediate Memory -- 32M words (65k DRAM)	200 (508)	130 (330)	76 (193)		140K (35K)	(1)
2	FMP Backing Store 65M words 256k CCD	65 (165)	29 (74)	76 (193)		26.8K (6.8K)	(1)
22	Prog. Device Controller	18 (46)	13 (33)	18 (46)		1.2K (0.3K)	
5	Condensing Unit 30 ton	90 (229)	34 (86)	48 (122)	40	290K (73K)	(2,3)
4	Condensing Unit 5 ton				7	55K (13K)	(2,3)
3	Motor-Gen 250 KVA	93 (236)	41 (104)	43 (109)	160	133K (34K)	
3	MG Control	56 (142)	20 (51)	78 (198)			

Notes for Table 2

1) The Freon-cooled units all dissipate about 10-15% of their total power into the room ambient. The balance is dissipated into the Freon refrigerant. The portion dissipated into the Freon is included in the total for the condensing units.

2) The condensing units dissipate about 5% to the room ambient and the balance to water. The total heat dissipated includes internal losses as well as heat taken from the Freon system.

3) The condensing units are normally in a separate room. The maximum length refrigerant lines are 100 ft (30 m).

NASF Physical Requirements Summary

- Total input power of approximately 1000 kVA.
- Total floor space requirements of approximately:
  - 6250 square feet for Computer Room.
  - 450 square feet for Compressor Room.
  - 400 square feet for Motor-Generator Room.
- Total heat dissipation of approximately:
  - 715,000 BTU/hr (180,000 kcal/hr) to Computer Room ambient air.
  - 115,000 BTU/hr (29,000 kcal/hr) to Compressor Room ambient air.
  - 445,000 BTU/hr (112,000 kcal/hr) to Motor-Generator Room ambient air.
  - 1,775,000 BTU/hr (447,000 kcal/hr) to water.

DIVISION 11

SYSTEM SIMULATOR

SUMMARY AND RESULTS

DIVISION 11

SYSTEM SIMULATOR

SUMMARY AND RESULTS

1.0 DEFINITION OF SYSTEM TO BE SIMULATED

The Numerical Aerodynamic Simulation Facility (NASF) as studied by CDC is a set of five (5) stations tied together by a communications trunk system called a Loosely Coupled Network (LCN). Figure 1 shows the generalized system.

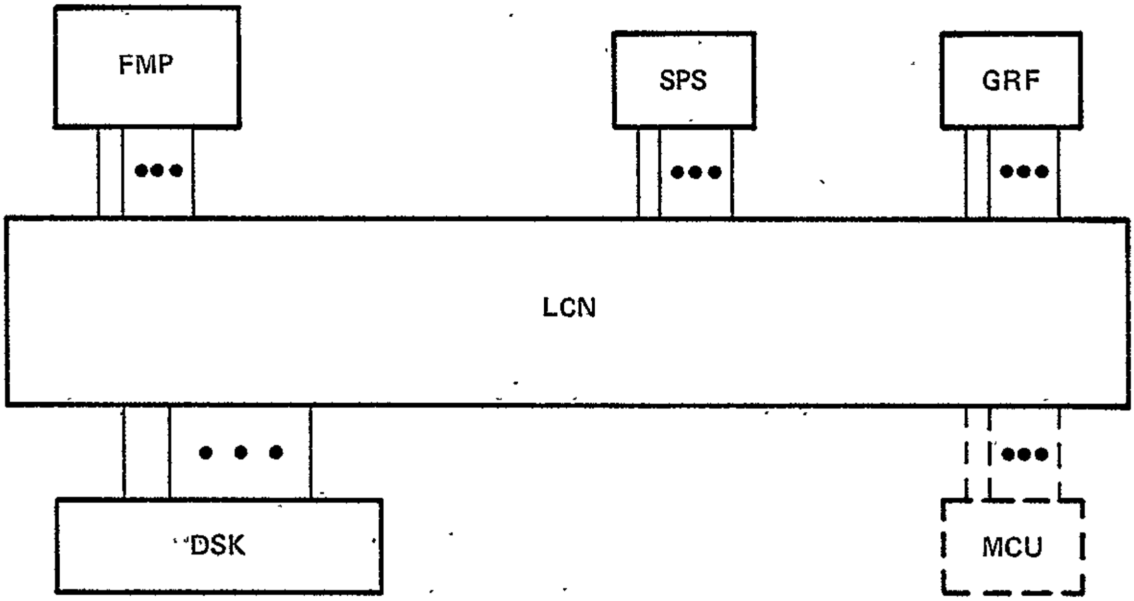


Figure 1. Generalized NASF

One constraint has guided the design of the proposed NASF system: to meet the user's computational need a computing engine (FMP) should be dedicated to executing flow code at a sustained rate of 1 billion floating-point operations per second (gigaflop). Thus it seems obvious that significant portions of the problem need to be carried out elsewhere. Typical job runs have been dissected to find basic functions lending themselves to modularization. With the job types so specialized and similar, such modularization can result directly in corresponding modular hardware. Jobs may now run "in parallel". When a job occupies a given module, say the FMP, the other modules are busy working on other jobs (i.e., setup and post-processing tasks). With proper balancing of module tasks the FMP execution can "hide" all other station functions. If this is true, the FMP would be 100% utilized. Job throughput would equal FMP throughput (neglecting the system's startup to steady state).

Now in addition to improved throughput, the distributed module concept allows designers to make some or all of the stations particularly useful entities on their own. In other words, aside from increasing processing efficiency and bandwidth, modules can provide storage capability, workstation processing, and/or significant CPU power, etc.

Though these principles were formed on the assumption of a specialized job, the hardware modules can match an apparently general job. For example, the following run -

a) job set up

   1) input setup
   2) code compilation
   3) pre-processing manipulation of data to generalized coordinate space
   4) transfer load module to computing engine

b) execution of code and storage of memory snapshots

c) post-execution functions

   1) store temporary results
   2) store long term results
   3) process result files for analysis
   4) produce graphical files

can be satisfied by a very versatile set of stations.

These hardware modules have been assigned the following rather canonical functions:

a) The SPS is the system controlling processor. It interfaces with the user community as a timesharing device with a powerful processing unit, and sets up and post-processes jobs for the FMP. It is expected that such a device will require the capability of between one and two times that of a CDC 7600.

b) The DSK serves as the data transfer pipe between stations. These disks serve as temporary storage and large buffer for the FMP engine.

c) The FMP is the computation engine for flow code or other lengthy set of calculations that the SPS cannot reasonably handle. Code compilation and data manipulation may also occur here.

d) The GRF is the display station for user interaction. Both graphical input and output are manipulated here. Limited processing power is available locally, plus a tie to the system network.

The actual hardware contents of these modules depends upon many things: job sizes, turnaround needs, storage requirements, etc. Simulation then becomes a powerful design tool: a workload is outlined from a job scenario, the parameterized system model then simulates the job. Simulation results may then be used to determine improvements to the hardware or software. Hopefully such a process converges to a reliable design.

1.1 Stations

A station, or device class, is a set of hardware (processors, disks, memory, etc.) which acts as a functionally unique unit within the system. The classes are:

1) Support Processor System (SPS)
2) Flow Model Processor (FMP)
3) Remote Storage Disks (DSK)
4) High Speed Graphics (GRF)
5) Maintenance Control Unit (MCU)

Details of the stations as presently proposed follow.

Note: The MCU is not included in the simulation, and will not be referenced again in this simulator discussion.

1.1.1 The SPS

The SPS represents the general purpose manager of the system. Composed of two CDC CYBER 175/116 computers and supporting an extended set of peripheral devices (local disks, communication controllers, mass storage system, etc.), it controls the flow of jobs among all stations. Basic responsibilities assigned to the SPS are:

1) Control and execute NASF operating system.
2) Compile FMP code.
3) Massage grid and configuration data for the FMP.
4) Process result files for display output.
5) Supply intermediate and long term storage.
6) Control all LCN file transmissions.
7) Timeshare local users.
8) Interface remote users to NASF.

One CDC CYBER device acts as the leader, holding and managing the mass storage directories. Otherwise each mainframe acts as a stand-alone processor working on its private job load. The CDC CYBER devices must contend for shared local hardware. Specifications for the proposed SPS are summarized below. Note that word sizes (in the SPS) are 60 bits.

1) Two mainframe processors, each with 262K words of central memory, 20 PPUs, 24 I/O channels, and 3-MFLOP sustained computation rate.

2) Four FMD disks--0.5 billion words total, 6-Mbit/sec transfer rate sustained. Each disk cabinet contains two disk spindles.

3) ECS--0.5 million words.

4) MSS--8 cartridge storage units, each with two cartridge tape transports, holding a total of 16 billion words active, cartridges removeable, seconds access time, 16-Mbit/sec transfer rate.

5) 12-Mbit/sec effective channel rate to LCN.

The main responsibility of the SPS station is to keep the FMP well fed with new jobs. All computations done in the SPS, e.g., code compilation, coordinate space transformations (grid and patch data), post-processing, are done with the intent that the FMP need not be weighted down with extra work. In addition each job can spend a large part of its processing or active life "near" the user. This is important for the interactive user both during code and input debugging, and post-processing analysis.

1.1.2 The FMP

The huge computational load of the fluid dynamics problem (or other numerical tasks which lend themselves to vectorization) is assigned to the FMP. With an estimated computation rate of 1 gigaflop sustained, keeping the FMP waiting for other stations is expensive. Up to 14 I/O channels (each 50 Mbit/sec) may be connected to the LCN. The FMP acts as a slave to the SPS and has no system responsiblities other than:

1) receiving files from DSK,
2) job computation,
3) sending results back to the DSK station.

The FMP might also have the capability to compile source code and do complicated pre-flow-code processing tasks. For example, some grid generation computations are too lengthy to assign to the SPS. In fact, simulation may show that the FMP has time for considerable non-flow-code processing, if the SPS is too heavily loaded.

1.1.3 The DSK

The DSK acts as a storage buffer available to all system stations. The sixteen 819-21 disks presently proposed provide 1 billion words of storage, or a total of about one day's worth of input/output for the FMP. Thus as a relatively large, fast access, fast transfer storage station, the DSK's responsibilities are to:

1) act as a backup queue to the FMP;
2) hold snapshots of intermediate solutions of large problems;
3) hold runs for possible same day restart;
4) transfer files to/from all other stations;
5) relieve other stations of current jobs' data bases.

The 819 disks are assigned to a number of dual-headed disk controllers (two channels per controller). Each half of a controller acts independently (with interlocks) allowing for increased bandwidth.

1.1.4 The GRF

The GRF station allows visual setup and solution assessment at a sophisticated graphics terminal. These terminals are tied into individual minicomputers with processing capability. Several of these terminals are concentrated together by a higher bandwidth computer. Each concentrator is a GRF device with one channel interfacing the LCN. The GRF responsibilities are to

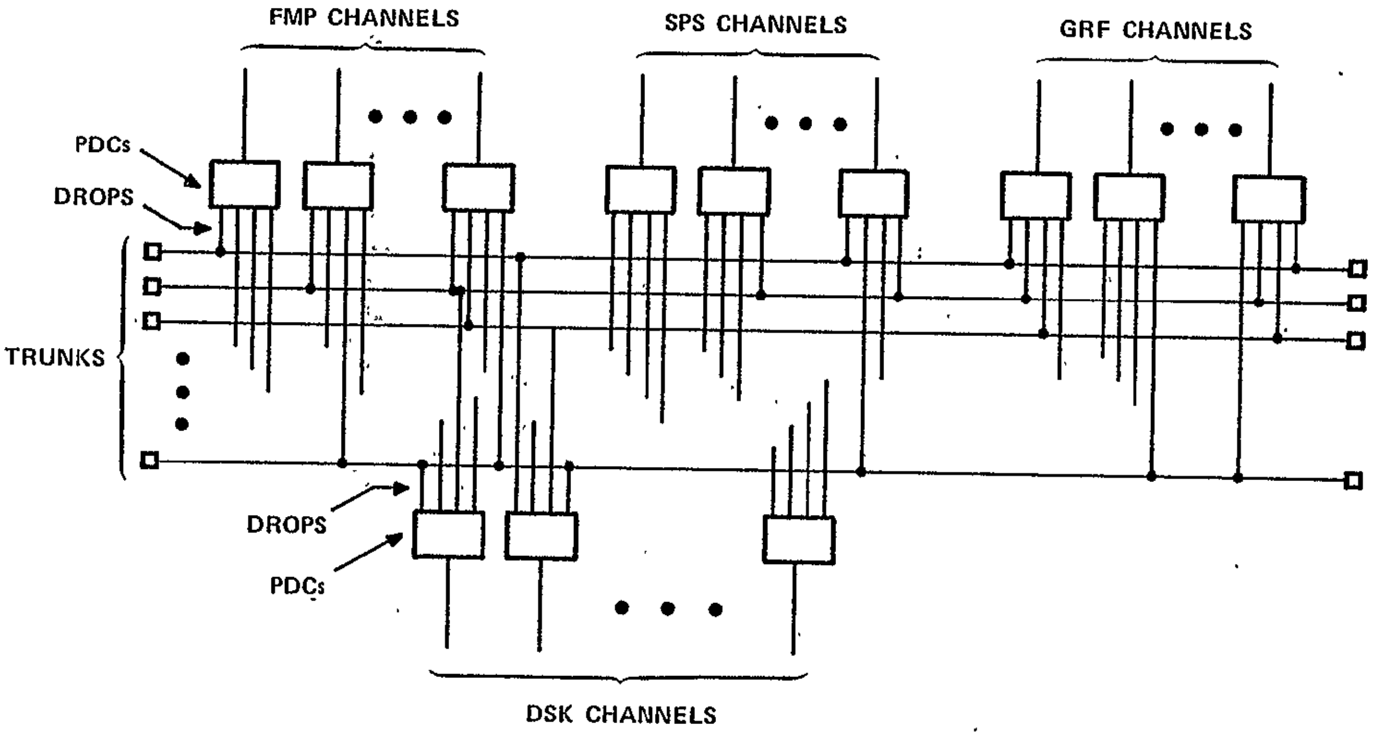
1) allow user construction of configurations (aeronautical test vehicles or parts thereof);
2) allow user to specify and construct flow fields;
3) analyze post-execution graphics frames;
4) be used interactively in the system;
5) transfer data to/from SPS and DSK.

Because this class of computer is much smaller than that of the FMP or SPS, data is sent in quarter-blocks using less of its central memory instead of full, 32K-word blocks (64-bit words). This station is intended mainly for the interactive aspect of the aerodynamics input preparation/result analysis.

1.2 The Loosely Coupled Network

The Loosely Coupled Network (LCN) is a communications scheme which allows both large data bandpass and versatile station to station connectability. The LCN centers on a number of bit-serial data trunks each with 50 Mbits/sec transfer capability. Programmable Device Controllers (PDCs) communicate with each other via the trunk system. Each PDC can connect with up to four trunks. (each connection to a trunk by a PDC is called a drop.) Each PDC connects to one device (via the PDC's backend). In this way a geometry configuration can be set up to suit nearly any designer's system philosophy. Figure 2 summarizes the hardware involved.

The responsibility of the LCN is straightforward, i.e., to handle all the communications load within NASF. For the job load involved, this is a major task. The usage scenerio dictates typical runs of very large data base jobs originating away from the computational engine (the FMP). In many cases significant portions of a job are passed between the GRF and SPS for setup or post-execution analysis. Long-term storage also resides far from the FMP in keeping with the FMP philisophy. Thus, to meet turnaround requirements and to keep the FMP "well fed", the LCN must be somewhat robust. For a detailed description of the LCN hardware please refer to Division 4 of Volume II.



NOTE:

ONE TO FOUR DROPS PER PDC
A MAXIMUM OF 16 DROPS PER TRUNK

Figure 2. NASF Loosely Coupled Network

1.3 System Philosophy, Principles, and Groundrules

The NASF is a distributed system with major functional capability in 4 discrete stations. The basic principle of using this design effectively is to perform parallel operations in these stations, i.e., simultaneous use of large amounts of hardware.

Processing may be done simultaneously in the FMP, SPS, and numerous stand-alone workstations. Parallel data transfers can potentially include

FMP <---> DSK
SPS <---> DSK
GRF <---> SPS
GRF <---> DSK

such that from 1 to 12 (if the LCN has 12 trunks) transfers can be moving on the LCN at the same time; six transfers at a time, for an effective bandpass of about 1.2 x 10^8 bits/sec, would not be unusual during peak demand. It should be noted that data is organized into large blocks, matching the block structure of the FMP Backing Store. Presently, such blocks are envisioned to have 32,768 64-bit words (plus SECDED).

A simple groundrule which allows this effectiveness is the lack of a central supervisor or controller on whose word everything awaits. Instead, each station governs its own action via its private "operating system". There is no need for one station to know what is happening in any other station. Data transfers between stations are preceded by short request messages. The FMP relies on messages from the SPS as to file locations and destinations but the operations are asynchronous and independent. Therefore devices, designed to be independent, see only their respective PDC channels and the messages which go through them.

System control, which resides in the leading SPS device, assigns and schedules jobs as they enter the NASF. After this point the job flows from one station to the next when the job is ready to proceed and the receiver permits it. The control is clearly not centralized. A system such as this, with much concurrency and capability, can still be overloaded by huge data base problems all crashing in the FMP., or mean loads whose data transfer characteristics change suddenly. In principle the system can be configured to handle nearly any workload as long as the usage is reasonably understood and generally predictable. In practice it would be enviable to have a workload which has an assortment of data transfer-processing time characteristics spread out uniformly through the day (i.e., the prime shift). In any case, the more accurate the usage model, the more appropriate a resulting system configuration is likely to be; and, the more simulation at different workloads, the more reliable a system design.

System reliability against breakdown is the last basic groundrule for the NASF, though not the least important. With significant amounts of hardware in each station and the vitality of each to the whole system, hardware redundancy is required. Except for the FMP, each station necessary for system operation has redundant devices, e.g., 2 CDC CYBER 175s, 4 disk controllers, etc. LCN breakdown must not be fatal to the system. Redundancy of device-to-device connections through the LCN can easily be three deep. The connection geometry must be designed to cover any single hardware failure (device, trunk, PDC, drop), and will, in fact, usually cover several multiple-breakdown conditions.

The loss of one support processor, 2 disks, and a PDC, for example, should not exclude the use of a healthy FMP. A good system design must continue to service the users, though perhaps at a diminished effectiveness, under such conditions.

2.0 SIMULATOR CHARACTERISTICS

The system simulators (SYS82 and SYS83) are written in a traffic-oriented simulation language, General Purpose Simulation System V (GPSS), in which transactions are serviced by the functional blocks through which they pass. Statistics may travel with the individual transactions; they may be compiled within each block; and/or they may apply to the whole system varying only with the flow of time. GPSS is appropriate here because 1) the programs can be developed to nearly any desired degree of detail and 2) the language is a standard simulation tool that can be used by Ames or nearly anyone with a CDC6600 class computer. Simulation is non-continuous at time resolutions near 1 clock period or less. For system simulation, a clock unit of 10 microseconds is used. This simulation should appear very smooth and continuous to the user, accustomed to experiencing the passage of time in seconds. Details of how to use the simulators are found in Volume IV, Division 1 and 2 (Simulator User Manuals); examples of using them as an analysis tool are shown in sections 3 and 4 below.

2.1 Code Structure

2.1.1 Central Theme - LCN Mechanics

The GPSS source code is centered on the operations of the LCN. The code is generalized so that the number of trunks, PDCs, and drops per each PDC, and the network connection scheme are all parameterized. These parameters are initialized at the beginning of the run and cannot be changed during simulation.

A message transfer begins by entering the sending PDC if available. The PDC, probably one of many on a given trunk, waits its turn and then seizes the trunk. While the trunk is taken, no other PDC can use it. After the message is sent on the trunk, the trunk is released. Before the trunk is dropped the receiving PDC answers "message received", "busy", or nothing. No reply is sent if the PDC is out of operation or if there was an error in the message transmission. Messages receiving busy responses are retried next time around unless preempted by a higher priority message in the sending PDC. Transmissions that receive no reply are tried several times and if still unsuccessful the system is notified of a perceived hardware breakdown. This should not occur in simulation, though statistically it could.

Heavy loads may create a queue of messages for a given sending PDC. This queue is ordered according to message priority, and is first-in, first-out within a given priority. GPSS "keeps the books" for the queues, and of hardware contention; no accuracy is lost as the run becomes complicated (though CPU time can grow).

PDC clocks are simulated at 10-microsecond increments. This represents the typical time increment for the internal PDC pointer that counts whose turn it is to take the trunk. If a particular trunk goes inactive, the appropriate clocks are turned off until needed, saving GPSS a lot of needless simulation effort.

2.1.2 Code Modules for Device Classes

Each device class is coded as a module. The modules contain a library of functions and capabilities, which define the internal workings of a class. This includes reactions of a class to the requests of other classes.

The DSK, FMP, and GRF modules are single, or nearly single, function stations.

The DSK stores and transfers data. The simulation module allows for one to eight dual-head disk controllers, each with a stipulated number of attached disks. Disk specifications are initialized at the beginning of the code. Each disk must be reserved for a specific transfer and then will allow only that data transfer. Other requests will receive "function busy" responses until the awaited disk transfer is completed. Contention for the disks can get quite complicated for heavy loads. The "disk access time" graph and table (see appendix A, pages 123 and 131 respectively, for examples) are sensitive indicators of the level of contention.

The FMP receives jobs, processes them, and returns them to the DSK. Limited communications between the FMP and SPS are also carried out. This module sets up the parallel transfers to/from the DSK. Maximum parallelism is exercised. This means if there are three I/O channels between the FMP and DSK, file transfers are split up into three groups. The FMP processor facility is monitored automatically by GPSS. A queue buffers jobs awaiting processing, i.e., a backing store. Only one job is processed at a time.

The GRF acts very nearly like the FMP. It transfers files and can be seized for processing. The GRF can transfer files to/from the SPS in addition to the DSK. In processing mode it can timeshare among users. The main difference is that a block transfer is done in pieces instead of all at once. This is simply because a 16-bit GRF concentrator most likely cannot hold an entire data block in its memory. The size of the transfers are stipulated and appropriate transfer times calculated. There can be from 1 to 4 GRF concentrators.

The SPS code is not nearly so modular. The SPS functions mingle with the system's operating system, and so characteristically the respective code fills the spaces between the LCN code and the stations' code. The basic functions are modularized: seizing a percentage of a mainframe processor and setting up data transfers to the GRF or DSK. The details of the data transfers are carried out in the "driven" devices, the GRF and DSK. Control responsibilities which permeate the entire source code include all look up directories (e.g., message path connections, disk allocation directory, etc.) and assignment of jobs to specific hardware devices.

2.2 Macroscopic Assumptions: Special Notes Concerning the SPS

Contention for shared hardware within the SPS is not modeled. Such a model is a separate project that eventually needs to be done. In the meantime the results of such a simulation has been estimated (carefully guessed), and used for input to the general system. These are called macroscopic assumptions. A four parameter family then becomes the performance capability of the SPS station.

1) Flop (computation) rate--This is the effective computation speed of each processor. For CYBER 175s a rate of 3 Mflops is assumed. Compilation speed is also included here. An accepted speed is 100,000 lines per minute.

2) Effective channel rate--The PPUs have well known data rates, but under a scheme of shared local disks, ECS, and MSS, determining the effective channel rate for large data blocks is not an easy problem. The contention scheme is comparable in principle to the total system contention scheme modelled in the LCN and DSK. The PPU I/O rate is assumed to be the effective channel rate; thus it is assumed that most of the overhead will be hidden.

3) Program or task load--Each SPS processor is a time sharing device. The order of task servicing is a complicated question. Jobs get rolled in and out depending on job memory and processing needs and a list of other state-of-the-system (SPS) factors. The macroscopic assumption is vastly simplified. Parameterized separately is the continuous load due to timesharing (workshop) users, load due to a data block transfer, and load due to program compilation or execution. These loads are percentages of each mainframe's capability. For example, the simulations in section 4 assign 20% for continuous workshop and operating system load, 10% for data block transfer load when such transfers occur, and from 15% to 70% for the various compilation and execution loads. Demand on memory is the main factor used in estimating these loads.

4) SPS response time--this represents the time it takes the SPS as a system to respond to internal device requests, and to communicate the external requests to its internal hardware. Thus, it is a macroscopic representation of SPS overhead time to requests and commands. It has not been determined if such overhead is of a significant time span and has been assumed to be zero.

Clearly the simulation results rest heavily on these assumptions. If later, more detailed SPS simulation shows estimates to be incorrect it is reasonable to expect that changes could be made in the SPS design to compensate.

3.0 THE SIMULATOR AS A TOOL

This section describes the different levels at which the simulator can analyze a system configuration alternative.

3.1 Simulator Input

Three sets of data can be manipulated.

1) The system hardware, via a deck of card images, as described in the User Manual (Volume IV)--this deck allows the user to describe communications between stations, network versatility, and hardware duplication.

2) A set of input commands, another card deck, which represents a job workload--each system job is constructed from the following commands:

    a) Transfer a file of blocks.
    b) Send a very short message (≈20 usec).
    c) Send a random length message (0 to 1000 usec).
    d) Seize FMP for processing.
    e) Seize a percentage of SPS or GRF device for processing.
    f) Change priority of input which follows.
    g) Assign arrival time for the next job of the same class.

    Typically from 10 to 20 input commands describe each job. Up to 99 jobs can be input in one simulation run.

3) A set of system parameters found at the beginning of the GPSS source code:

    a) Hardware parameters - DSK specifications, channel rates, buffer sizes within the PDCs, and guidelines for hardware duplication.

    b) Software and firmware assumptions - block size, interstation block transfer time statistics, workshop load on SPS and GRF devices, load demanded from SPS and GRF for block transfers.

Data sets 1) and 3) represent a system design to be tested. The user should be careful to understand that the simulated results represent the input designs and assumptions made in these data sets. Thus a large responsibility rests with the user: to understand and note the assumptions made, and to acknowledge the inherent shortcomings and analytical shortcuts that result. For example assigning one number, say 10%, to represent the load put upon an SPS processor to transfer a 32K-word data block is an oversimplification. The sensativity of the results to such a macroscopic parameter should be understood. Extrapolation of results can only be suggested after several similar simulations.

3.2 Two Simulators

There are two levels of simulation detail (see Volume IV). Source code SYS83 is a general purpose simulator which studies the whole system. In this code, system functions are modeled macroscopically: block transfers, seizing processors for seconds, contention by several/many jobs of the various system resources. Functional detail in the SPS, FMP, and GRF is no better than tenths of seconds. Detail in the DSK station is at the millesecond level (seek times, rotation rates).

Source code SYS82 models message transfers in much greater detail. Block transfers are done at a higher level of detail, i.e., each sector is modeled. Buffering of sectors in the PDCs is performed when station channel rates do not match, e.g., CYBER 175 and 819 disks. Thus the detail of the transfers is nearly at the 10-microsecond level within the LCN. This simulator yields accurate macroscopic specifications used in SYS83 for block transfer times. In addition disk assignment algorithms within the DSK can be compared.

Thus though the two simulators are used in tandem, the responsibility of system analysis rests within SYS83.

3.3 Simulation Techniques

3.3.1 Full Run - Truncated Run

Simulation runs are performed in two modes.

Mode A: Simulation to completion of all jobs. This acts as a full master listing of an entire simulation. The system is potentially seen under several conditions: start up, smooth throughput, backed up, and clearing out at the end. Because of the random nature of job arrivals, the end of such a run can be greatly extended by a few jobs which arrived late. This situation distorts the mean statistics of the simulation -- in some case quite badly (20%). Yet such an overview of the run is a good start.

Mode B: Simulating until an initially stipulated number of GPSS transactions have been sent. This allows the user to stop the simulation while the system is in a specific condition. For example, with the help of the first master run (a Mode A simulation), the user can stop the simulation at virtually the minute chosen -- perhaps after 85% of the jobs have run and another 10% are at various stages of completion. In this case, the mean statistics are more valid. See User's Manual (Volume IV, Division 1) for implementation.

In some cases, there may be an anomaly in stopping the system in full flight in that some "facility" output may be incorrect, i.e., "average utilization" and "average time per transaction" for the FMP and SPSs may be wrong. In this case a simple hand calculation from the individual job histories should yield both the average sieze time per transaction (A) and the number of transacations (N). Then, with the simulator stop time (S) from the chronological history, the facility utilization is easily determined.

Average utilization = (A*N)/S

A scan of the simulation history should make clear whether this correction is needed.

3.3.2 Light Load - Heavy Load

Experience has shown that the first step in an analysis of a given NASF design should be a set of simple problems. A general analysis involves an understanding of a complicated set of coupled resources and is simply not possible without first understanding some of the basic principles that come out of each system design. The light load - heavy load technique isolates the interconnection design. It assesses the potential of the trunk system and the conflicts therein.

A) Light Load

A light load is defined as a set of data transfers which take less time than the mean time between transfer requests. This can occur either because of a generally light workload, or because the workload is dominated by processing time. In these cases disk conflicts, PDC busy replies, and trunk busy replies are rare. The user sees the network reaction in best case mode.

The relative interstation bandwidths, trunk loads, and PDC loads show clearly. Communications balancing problems should then become evident.

Appropriate data for these runs are parts of the workload to be modeled later, e.g., Model #1 of the Ames Usage Model, alone. These runs can also be used to debug the workload data.

B) Heavy Load

A heavy load is a workload dominated by data transfers numerous enough to saturate at least part of the LCN hardware. For example take the above light load input and decrease all the job interarrival times. Communications hardware with average utilizations of 70% or more are usually considered saturated or partially saturated. If potential bottlenecks in the LCN were not apparent before, they surely show here. More importantly though, the system shows just how heavy a load it can stand. The user should note decreased communication efficiency through a range of loads.

At this point system reconfiguration may be deemed necessary. If so, the light load - heavy load technique is begun again. By the time this process is complete the user should have a good understanding of a proposed system.

3.3.3 Key Diagnostics

A completed simulation yields a history of all the jobs executed. File transfers, processing times, start job and end job annotations are listed. A wealth of information is provided in these listings. These provide a feeling for how the system progresses through its various states. This running time history helps the designer isolate particular resource demand problems. System wide software algorithms, e.g., job priority assignments, can be assessed. When all the information is digested, the simulation can be a powerful tool.

The general simulation results are summarized by a few key statistics. These can be scanned and digested easily.

a) FMP and SPS utilizations. Hopefully the utilization of the FMP is quite high which, as noted in section 1, is the whole idea of the NASF system. On the other hand, the SPS should not be over-saturated. Overhead time due to job conflicts would slow down the feeding and finishing of jobs; system throughput would diminish.

b) FMP and SPS queues. The FMP may build up a significant job queue depth as long as the depth is not monotonically increasing in time. A steady state FMP queue will not slow down throughput. This is not true of the SPS queues. Such a queue represents an inability for the SPSs to keep up with the system demands; thus slower system startup, finish, turnaround, and throughput.

c) Trunk utilization. This should show if the LCN had any trouble keeping up with the message and data traffic workload.

d) Disk access time. The greater the number of cylinder seeks required, the more the data transfer overhead, i.e., the less efficient the DSK station. Such seeks are due to file request conflicts to the disks, a characteristic of a heavy communications load.

e) Mean block transfer time. Again this shows overhead due to heavy data transfer loads. In this case transfer time increases as a block awaits an open line on the network. File transfer times also show backups in extreme cases.

f) Throughput statistics. This is the bottom line of performance to every simulation. Did the system finish jobs as quickly as it gets them?

3.4 Trial Runs on Three Different LCNs

Three different NASF configurations, all with the same device class hardware, have been tested. Figures 3, 4, and 5 show the respective connection schemes for LCN1, LCN2, and LCN3 respectively. Trunks 1 through 4 are dedicated to FMP <--> DSK transfers (if 4 are used). This bandwidth is a response to the principle that a pipe to/from the FMP always be clear. This includes the peak loads of checkpoint dumps, typically millions of words. Trunk 5 is for the high priority SPS <--> FMP messages and SPS <--> GRF data transfers (relatively rare). Trunks 6 and 7 (when used) are for SPS <--> DSK and GRF <--> DSK data transfers. Clearly not all the network hardware is used in all three cases. The differences are simple: LCN2 has twice the SPS <--> DSK and GRF <--> DSK bandwidth of LCN1. LCN3 has the same SPS <--> DSK and GRF <--> DSK improvement, but half the FMP <--> DSK bandwidth.

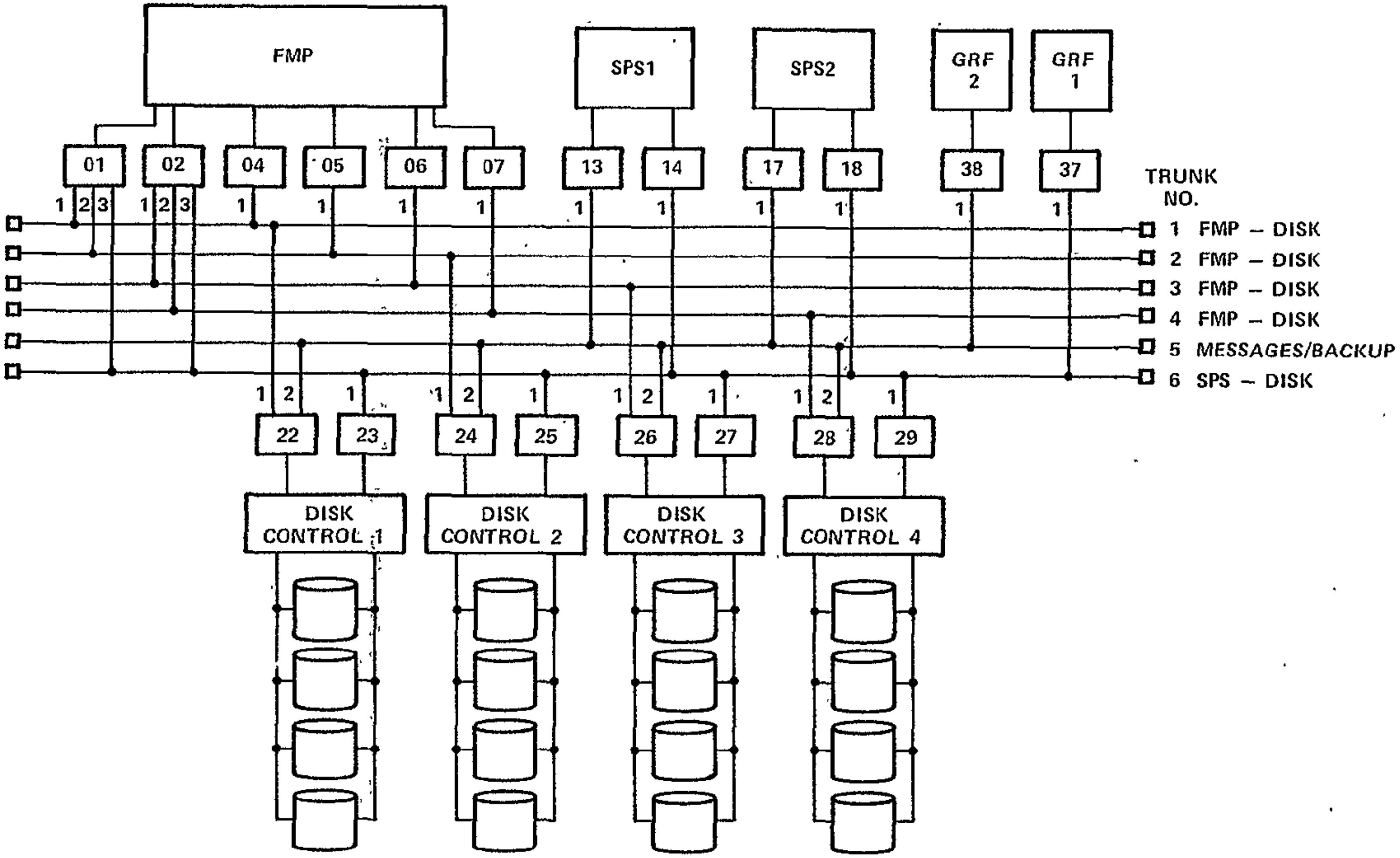


Figure 3. LCN1

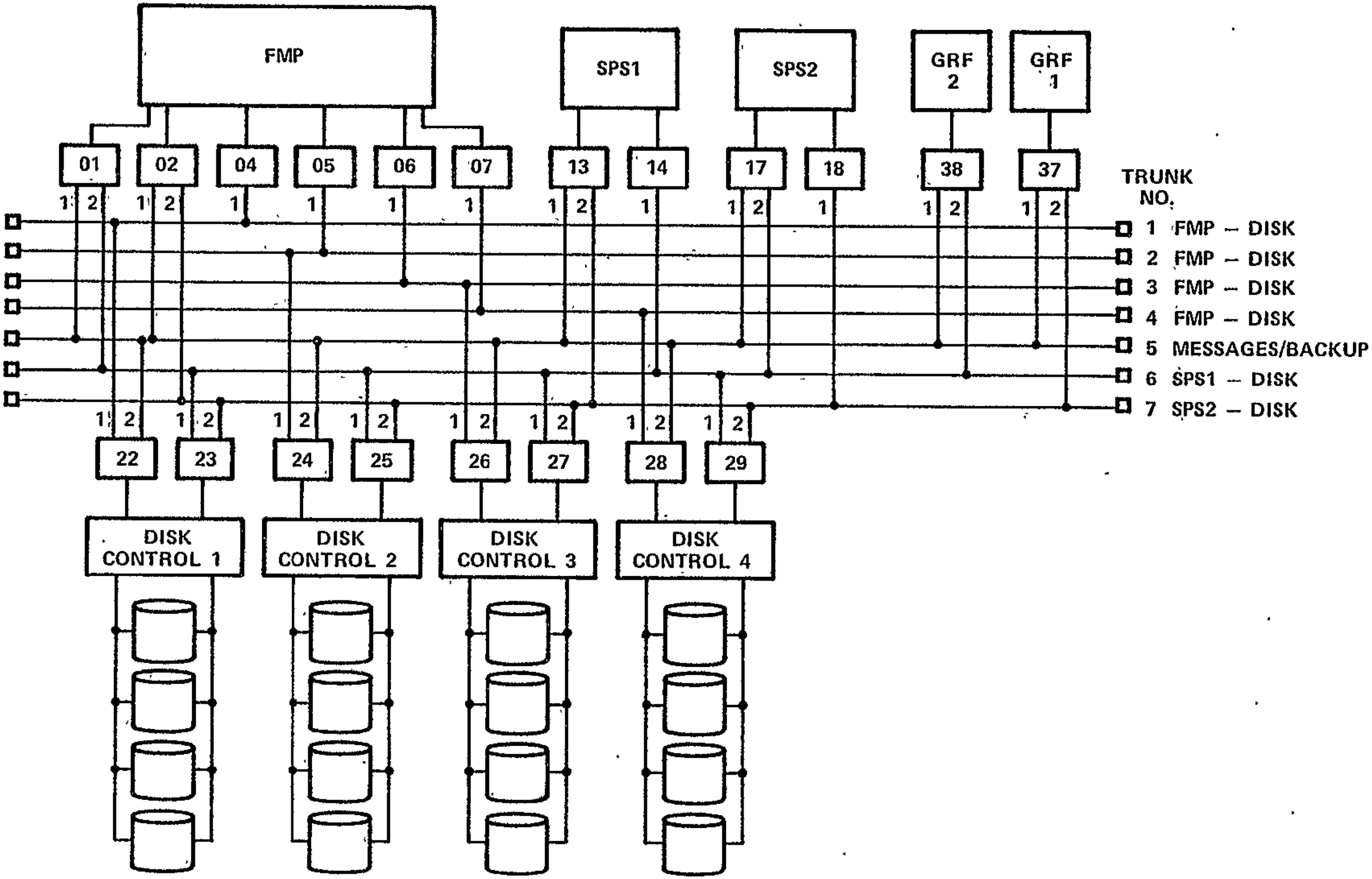


Figure 4. LCN2

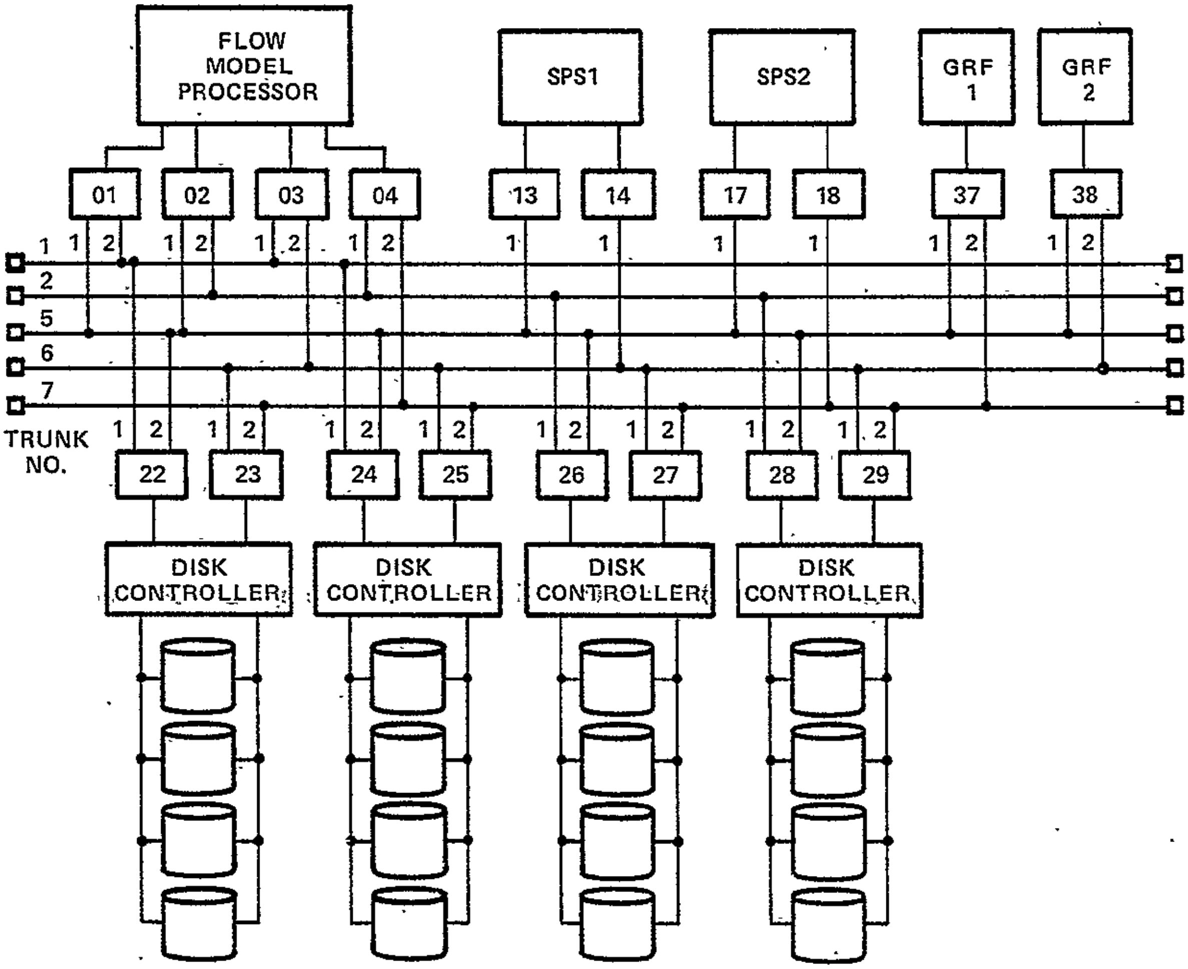


Figure 5. LCN3

Simulation assumptions:

Block size: 64 sectors (each 512 words),
i.e., 32K words (64-bit)
CYBER PP rate: 12 Mbits/sec
Disk (819) transfer rate (sustained): 21 Mbits/sec
Trunk rate: 50 Mbits/sec
GRF channel rate: 2 Mbits/sec
PDC buffer size for data transfers: 3 sectors
% load on SPS device for data transfers: 10%
% load on GRF device for data transfers: 80%

LCN1

Under relatively light loads (input = 1.4×10^{8} words/hr) it was immediately obvious that the SPS <--> DSK bandwidth is not balanced with that of the FMP <--> DSK which is 5 times faster. Utilization of PDCs 14 and 18, the SPS's prime channels to DSK, is about twice that of all other PDCs. PDCs 1-7 and 21-29 and trunks 1-4 all have very comparable utilization averages.

Under heavier loads the SPS <--> DSK imbalance relative to the FMP <--> DSK grows, until trunk 6 begins to saturate at around 70%. At this stage the SPSs are utilized 19% and 14% respectively. With mean block transfer times doubled, the SPS station is clealy I/O bound by trunk 6. At 4.8×10^{8} word/hr all LCN hardware is less than 50% utilized except trunk 6 (71%). This data load exceeds the needs stated in the Ames Usage Model. Though trunk 6 is almost saturated the effective transfer rate is well within data throughput goals.

LCN2

Now trunk 6 and 7 share the SPS <--> DSK and GRF <--> DSK load originally on trunk 6. Now SPS <--> DSK is only 2.5 times slower then FMP <--> DSK. This alleviates much of the excessive load seen on trunk 6 of LCN1. Otherwise system balance is identical to LCN1. At an input load of 4.3×10^{8} words/hr (output = 1% of input; FMP processing times = 60 sec/job Usage Model-Model#2), no part of the network has begun to saturate! Trunk 6 is about 50% utilized; trunk 7 is about 30-35% utilized. SPS utilizations have dropped to 7% and 3% respectively. Job throughput and turnaround times improve by about 10% for heavy workloads.

LCN3

Cutting the FMP <--> DSK bandwidth in half (otherwise identical to LCN2) does not degrade system throughput or job turnaround at all. The LCN hardware now looks well balanced with the FMP and SPS having comparable bandpasses into the DSK. Even so, extra system reliability, versatility, and the possibility of larger checkpoint dumps prompt a preference for LCN2 over LCN3 though at slight extra cost.

4.0 RESULTS FROM SIMULATING THE "NASF USAGE MODEL"

Two to three hour slices of the NASF Usage Model (version 79.001, from NASA-Ames) have been simulated on the three proposed systems described in section 3.4. The multi-job, high-level simulator, SYS83 has been used.

Below, one example is presented in detail to show how simulation analysis may proceed. Several other examples are referenced in passing to help clarify basic points. It is important, though, to understand that these represent first passes at using simulation tools for the NASF. Much time and effort will be required for a complete analysis.

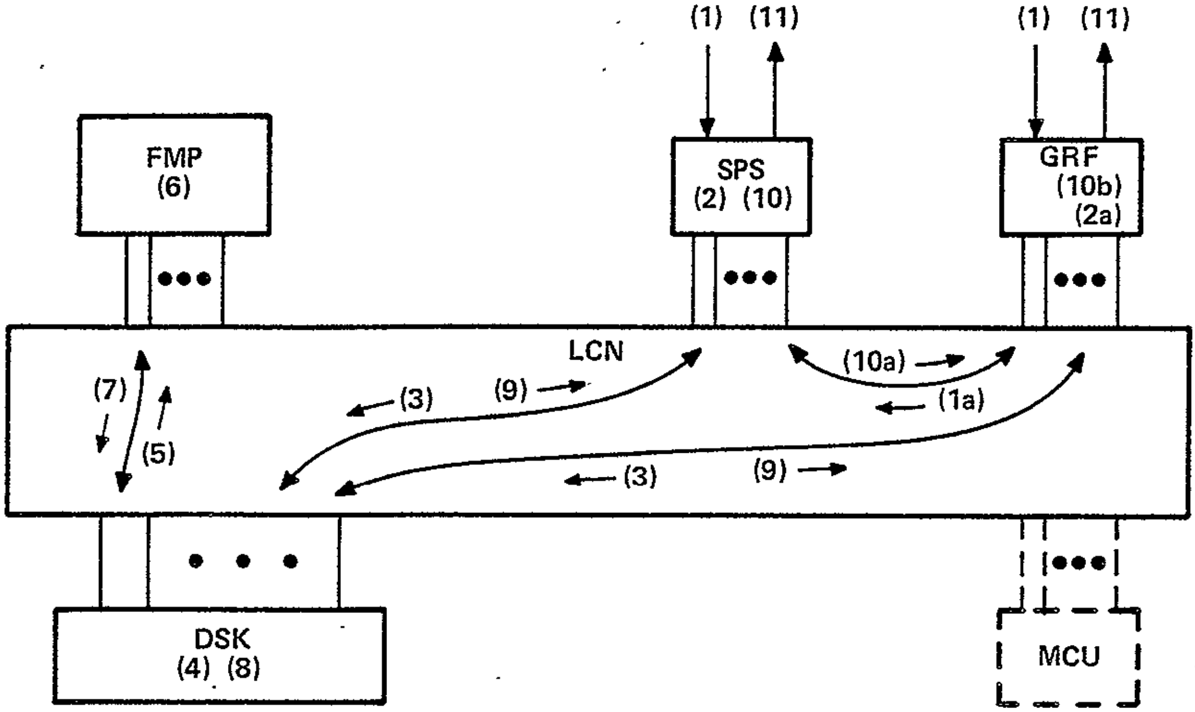
4.1 Translation of Usage Model into Workload Input for Simulator

The Usage Model translates fairly easily into simulation input. Model guidelines are followed as closely as possible with one exception. Model 4, "Complex Design Simulation", jobs are assigned entirely to the prime shift instead of 1/2 prime, 1/2 night. This choice is made with the intent of loading toward worst-case simulation.

Fifty-nine Model #1, twenty-three Model #2, six Model #3, and eight Model #4 jobs were simulated. This represents the average job load of 2 hours during the prime-time shift. Specifications are summarized below. Within each class of jobs the major commands that represent that class are listed with letter headings. For some letter headings job scenarios branch out into one or more variations. Such branching is represented by a multi-level number sequence added onto that letter (similar to section or paragraph heading numbers).

The letters represent major steps in a job flow. The first digit, if present after a letter, represents a branch or split of jobs in that class. The second digit, if present, indicates a sequence within the leg of the branch represented by the first digit.

The flow of a job through the system is actually very similar for all jobs. The variation is in the number and sizes of the files involved, and the processing time required. Figure 6 summarizes the basic scenario.



Notes:

(1) "Job execute" request enters NASF through SPS or GRF.
(1a) GRF files may go to SPS.
(2) SPS compiles source code and/or preprocesses input data.
(2a) Continue to (3).
(3) SPS and/or GRF files transfer to DSK.
(4) DSK stores input files; load moule and input data.
(5) Input files sent from DSK to FMP.
(6) FMP executes job.
(7) Result files sent from FMP to DSK.
(8) DSK stores result files.
(9) Some/all result files transfer from DSK to the SPS and/or GRF.
(10) SPS post-processes result files for user analysis.
(10a) Post-processing results may go to GRF for display/analysis.
(10b) Continue to (11).
(11) Job returns to user at workstation; further analysis may proceed on graphics hardware, a local processor, or the SPS.

Figure 6. Typical Job Flow through NASF.

Model #1 Method Development (59 jobs). These jobs have small processor demands (both SPS and FMP) but relatively large data bases. Typically this is a three million word job which runs in the FMP for 10 seconds, followed by the retrieval of a dump file or diagnostics.

a) Job execution request arrives; interarrival mean of 120 sec.

b) SPS compiles prepared source code. 30% of one CYBER mainframe seized for 1 sec; then

c.1) No wait - go to step d) (14 of 49 jobs).

c.2.1) Wait 10 minutes ("think time") due to compilation error and recompile as in b) (for 45 of the 59 jobs); then

c.2.2) No further wait - go to step d) (30 of 45 jobs); wait 5 minutes ("think time") due to recompilation error and recompile as in b) (for 15 of the 45 jobs); then

d) Send File1 (load module - 1 block) to DSK; then

e) Send File1 from DSK to FMP, and

f) Send File2 (configuration - 1 block) from GRF to DSK; then

g) Send File2 from DSK to FMP, then,

h) Request FMP for execution. The FMP checks for arrival of files 1 and 2 and processes them for 10 sec as soon as allowable.

i.1.1) Send File9 (debug dump file - 90 blocks) to DSK (for 20 of the 59 jobs), then

i.1.2) Send File9 from DSK to SPS (same 20 jobs).

i.2.1) Send File8 (edited results file - 2 blocks) to DSK (remaining 39 of the 59 jobs), then

i.2.2) Send File8 from DSK to SPS (same 39 jobs).

i.2.3) Request SPS processor for output/pictorial post-processing. Seize 25% for 40 sec (same 39 jobs).

j) End job.

Model #2 Code Development (23 jobs). Similar to Model #1, but the processor demand and data base is larger. Half of the jobs require post-processing for graphics use. Typically an eight million word job which may run in the FMP for 60 seconds.

a) Job execution request arrives; interarrival mean of 300 sec.

b) SPS compiles prepared source code. 50% of one CYBER mainframe seized for 1 second; then

c.1) No wait - go to step d) (6 of 23 jobs).

c.2.1) Wait 10 minutes ("think time") due to the compilation error and recompile as in b) (for 17 of the 23 jobs); then

c.2.2) No further wait - go to step d) (11 of 17 jobs); wait 5 minutes ("think time") due to recompilation error and recompile as in b) (for 6 of the 17 jobs).

d) Send File1 (load file - 2 blocks) to DSK; then

e) Send File1 from DSK to FMP, and

f.1.1) Send File2 (configuration data - 2 blocks) from GRF to DSK (19 of 23 jobs), then

f.1.2) Send File2 from DSK to FMP (same 19 jobs)

f.2.1) Send File2 from GRF to SPS (4 of 23 jobs), then

f.2.2) Request SPS device for configuration manipulation. Seize 30% of one CYBER for 70 seconds; then

f.2.3) Send File2 (resulting transformed configuration data - 2 blocks to DSK; then

f.2.4) Send File2 from DSK to FMP (same 4 jobs).

g) Send File3 (grid data - 19 blocks) from SPS to DSK, then

h) Send File3 from DSK to FMP.

i) Request FMP for execution; FMP checks for arrival of files 1, 2, and 3 and processes them for 60 sec., as soon as possible.

j.1.1) Send File9 (debug dump - 125 blocks) from FMP to DSK (for 8 of 23 jobs), then

j.1.2) Send File9 from DSK to SPS, and

j.1.3) Send the rest of File9 (125 blocks more) from FMP to DSK (same 8 jobs), and

j.1.4) Send this part of File9 from DSK to SPS.

j.2.1) Send File8 (edited result files - 6 blocks) from FMP to DSK (12 jobs of 23), then

j.2.2) Send File8 from DSK to SPS (same 12 jobs).

j.2.3) Request SPS for output/pictorial post-processing. Seize 70% of one CYBER for 120 sec. (same 12 jobs);

j.2.4) Send File7 (output file - 4 blocks) from SPS to GRF (same 12 jobs);

j.3.1) Send File8 (edited result files - 12 blocks) from FMP to DSK (remaining 3 of 23 jobs), then

j.3.2) Send file from DSK to SPS (same 3 jobs).

j.3.3) Request SPS for output/pictorial post-processing. Seize 70% of one CYBER for 240 seconds (same 3 jobs), then

j.3.4) Send File7 (output file - 8 blocks) from SPS to GRF (same 3 jobs).

k) End job.

Model #3 Simple Design Simulation (6 jobs). Engineer's simulation job. 20% of the jobs have full blown configuration and grid files. Some pre-processing short FMP run-60 seconds. 20% restart. Heavy post-processing for graphical use..

a) Job execution request arrives; interarrival mean of 24 minutes.

b.1) Proceed to step c) (5 of 6 jobs).

b.2) SPS compiles prepared source code. 70% of one CYBER seized for 2 seconds (1 of 6 jobs).

c) Send File1 (Load file - 4 blocks) to DSK, then

d) Send File1 from DSK to FMP.

e.1.1) Send File2 (configuration patch - 4 blocks) from the GRF to the SPS, (3 of 6 jobs).

e.1.2) Request SPS for configuration and grid manipulation. Seize 70% for 240 sec (same 3 jobs), then

e.1.3) Send File3 (transferred patch and grid data - 92 blocks) from SPS to DSK (same 3 jobs); then

e.1.4) Send File3 from DSK to FMP.

e.2.1) Send File2 (patch and grid setup - 4 blocks) from SPS to DSK (3 of 6 jobs); then

e.2.2) Send File2 from DSK to FMP (same 3 jobs); then

e.2.3) Request FMP processing of patch and grid manipulation. Seize FMP for 10 seconds (same 3 jobs).

f) Request FMP processing for flow code for 60 seconds.

g.1) No File9, go to step h) (3 of 6 jobs).

g.2) Send File9 (restart file - 220 blocks) to DSK for temporary storage (1 job of 6)

g.3.1) Send File9 (raw results - 150 blocks) to DSK (2 of 6 jobs), then

g.3.2) File9 from DSK to SPS (1 of 6 jobs).

h) Send File8 (edited results file - 3 blocks) to DSK, then

i) Send File8 from DSK to SPS.

j) Request SPS processing of file 8 for pictorial results. Seize 60% of one CYBER for 200 sec.

k) Null - reserved for Model #4.

l) Send File7 (display file - 2 blocks) from SPS to GRF for graphics study.

m) End job.

Model #4 Complex Simulation Design (7 jobs). Engineer's simulation jobs requiring significant processing time -- 10 minutes in the FMP, several minutes in the SPS for pre-processing or post-processing or both. This model has the heaviest total demand of time on the FMP (6.5 hrs/day).

Steps are identical to Model #3 except as noted below.

a) Mean job interarrival time: 15 min. (900 sec.)
f) Request FMP processing flow code for 600 seconds.
g.2) Restart file - 310 blocks.
h) Edited results file - 8 blocks.
j) Seize 60% of one CYBER for 240 sec.
k) Seize 60% of one CYBER for 240 sec. again.
l) Display file - 10 blocks.

The above represents two simulated hours of input during prime shift. Mean job interarrival times are based on the assumption that the prime shift is 10 hours long. Estimates for SPS processing times have been made for code compilation, pre- and post-processing tasks. The main difficulty is counting the number of calculations required. An estimate of compilation rates from past experience with CYBER 170 family is 100,000 lines of code/min. For grid and/or patch generation/transformation, 10^{10} calculations per 10^{6} points is assumed (quoted from Ames' Usage Model). For post-processing of result files 6×10^{7} calculations per 4000 point contour plot (i.e., one frame) is assumed (again from Ames' Usage Model). Each CYBER 175 is estimated to run at 3×10^{6} calculation/sec when the entire job is in its central memory. A 50% reduction in speed is assumed if only 1/2 the job is in the central memory at a given time.

At several junctures of the input design, estimates were assigned. Examples are 20% continuous load on each SPS mainframe for workshop timesharing, high estimates for SPS computation times, and more restart and raw result files than expected on average (by 60%). Thus, it appears likely that a heavier than average primeshift workload, is being simulated, though not an unlikely load.

4.2 An Example Simulation

A truncated run (Mode B), lasting 7650 seconds, is described here as a typical example of a system evaluation via simulation. LCN2 was used as the baseline configuration. The run follows all specifications listed in section 3.4 and section 4.1 with one exception: SPS compiling and processing times have been lengthened by 50% as a tradeoff for demanding half as great a processor resource load. For example, a 120 second job requiring 70% of an SPS processor is converted to a 180 second job requiring 35% of the processor. This allows much greater processing versatility within the SPS, but at the same time implies a much greater SPS computation power. Specifically, instead of a pair of 3-megaflop/sec machines as described in section 1.1.1, now a pair of 4.5 to 5-megaflop/sec machines are being modeled. This SPS "horsepower" parameter is varied throughout the family of simulations from this high rate down to a minimum of 2 megaflops/sec.

This simulation took approximately 700 CPU seconds on a Cyber 175. For a full listing of the results see appendix A. The amount of information contained in a run like this is quite large, though not unmanageable. With time it is reasonably digestible, as hopefully the reader will see in the following presentation.

4.2.1 Characteristics of the Run

4.2.1.1 Job Arrivals

Though job arrival specifications are quoted in the usage model, actual job arrival times have a random nature. Thus for a given run these arrival times are unique. Figure 7 shows the job arrival rate distribution throughout the 2 hour simulation. This distribution is plotted by taking mean arrival rates each 15 minutes. The mean of 20.2 jobs per 30 minutes is 16% less than the 24 jobs expected; but by far the most demanding job class, Model #4 "Complex Design Simulation", was fully represented. Perhaps a more characteristic number is the actual load demand for the FMP versus the expected demand: 93%. This is not to be confused with the FMP utilization. Arrivals of these demanding and important Model #4 jobs are marked by arrows in figure 7. Note - 1) the close arrival of two of these jobs near clock time 1100 seconds, and 2) the arrival of three of these jobs within 10 minutes of each other at 1 hour wall clock time.

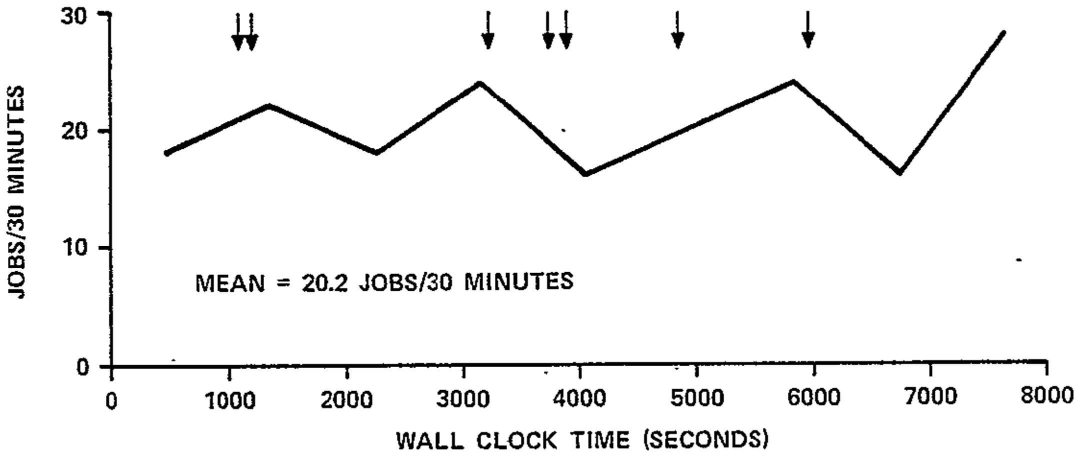


Figure 7. Job Arrival Rate

4.2.1.2 SPS Utilization

Figures 8 and 9 summarize the utlization of the SPS resources due to program compilation and execution. Three characteristics of interest are clear:

1) The spacings in time of general loads shows the results of the job to device assignment algorithm. This algorithm is quite simple: When a job enters the NASF it is assigned to the SPS device whose central processor and memory are least loaded. Under equal loads SPS1 wins the honor. So, in general, without deference to looking ahead, the SPSs ping-pong the responsibility back and forth. In this run the SPSs complement each other well.

2) The sharp upward spikes seen, especially in the SPS1 utilization graph, show the quick response it has to demands. The general lack of broad blocks along the time axis for large loads represents very good CPU availability. Arrows point to times when a job enters the processor after having been blocked out. They are rare and short lived.

   The continuous block of time at 20% represents the timesharing load.

3) The required load assumed for block data transfers, when they occur, is 10%. The near absence of SPS utilization greater than 90% shows that file transfers are rarely held up by the CPUs.

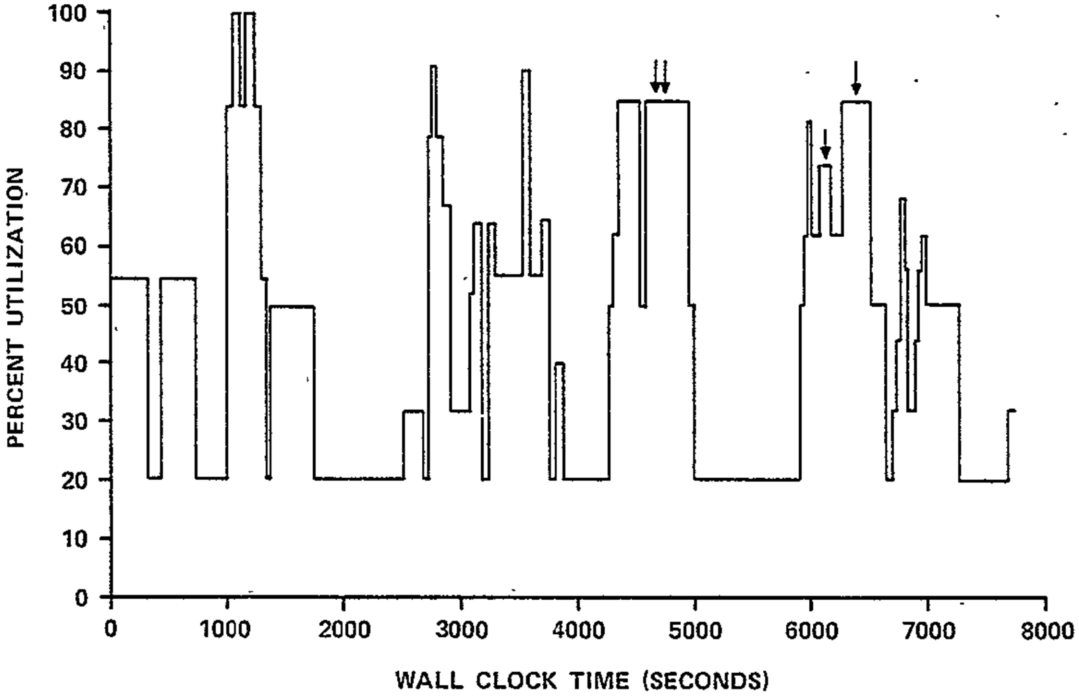


Figure 8. Utilization of SPS1.

Arrows represent jobs which were originally blocked out of the processor.

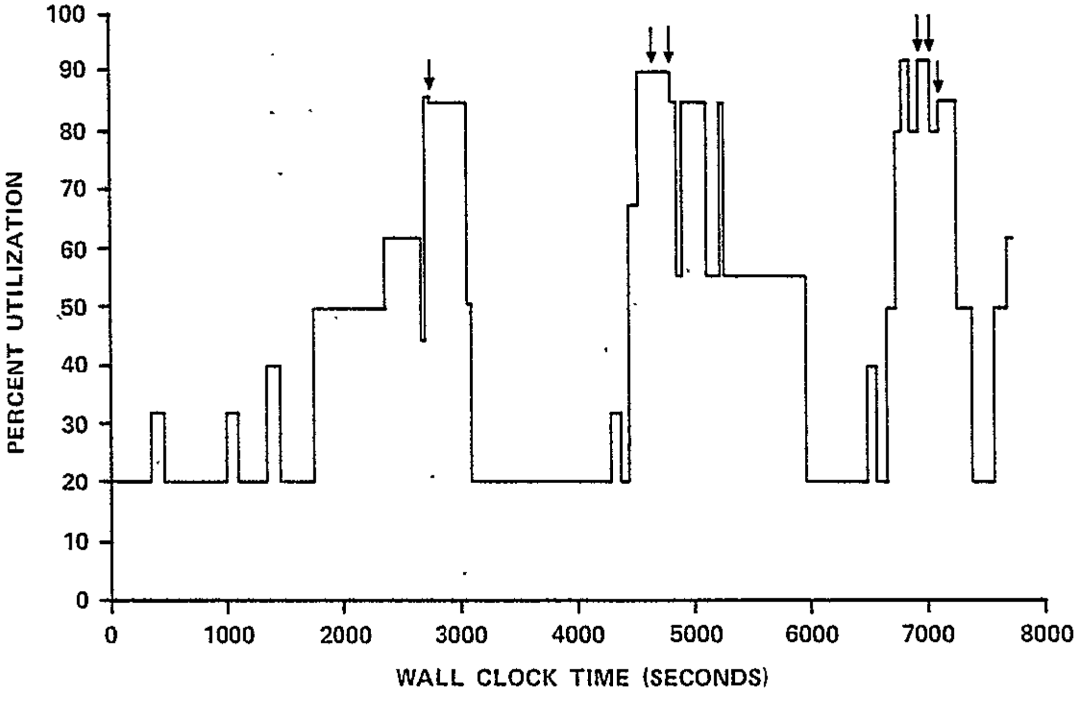


Figure 9. Utilization of SPS2.

Arrows represent jobs which were originally blocked out of the processor.

4.2.1.3 The FMP Execution Queue

The driving philosophy for a well-used NASF is to keep the FMP busy. A view of the FMP execution queue is a good diagnostic for seeing if the workload is going to adhere to this point. Figure 10 shows the queue for this run. In general, the queue is well fed with a mean depth of 5.33 jobs.

The graph shows that, on this job mix, the FMP can keep up with the job load by emptying the job queue several times.

The driving force in building up the FMP queue is undoubtedly the beginning of an FMP execution on a Model #4 job. Arrows in figure 10 show these execution start times. The executions are 10 minutes long, and can easily cause the queue to grow by 10 jobs. Notice also how these Model 4 jobs have now spread out from one another in time This is inevitable, though the actual spacings can be considerably widened or partially thinned by using the job priority cards mentioned in section 3.1.2.

4.2.1.4 FMP Utilization

As the reader should be able to predict from seeing the FMP queue results, utilization of the FMP processor should be relatively high. Utilization is, in fact, a healthy 86%. Please note figure 11. Startup time accounts for 5% of the unused resource; this overhead need not occur again during the day. Of the FMP time not used, 7% is due to the less than expected demand caused by the job arrival statistics (93%). Thus at "simulation stop" the FMP is about 7% behind. Such a number is somewhat sensitive to stop time; at 7000 seconds the number is about 4%. Only during the period from 5400 to 6700 seconds does the FMP load get "seriously" behind.

The reader should note that in appendix A (page 120) the FMP utilization statistic has been corrected by hand. This simulation run stopped in "full flight" (i.e., a truncated run) and thus the 46% FMP utilization printed seemed suspect. After checking individual job histories as described in section 3.3.1, this utilization was appropriately corrected. Under such circumstances other facility statistics must also be checked for consistency. In the case cited only the FMP utilization statistic was spurious.

4.2.1.5 Throughput and Turnaround Statistics

Figure 12, showing the throughput graph, reflects the results of the previous 5 graphs. In addition, it shows vividly the one discouraging point in this example: job turnaround times are neither consistent nor very fast. With the throughput rates varying as they do here, it must seem at times as though the system has gone to sleep. The fact is that at these lull times in the throughput (clock time from 4000 to 6000 seconds), the

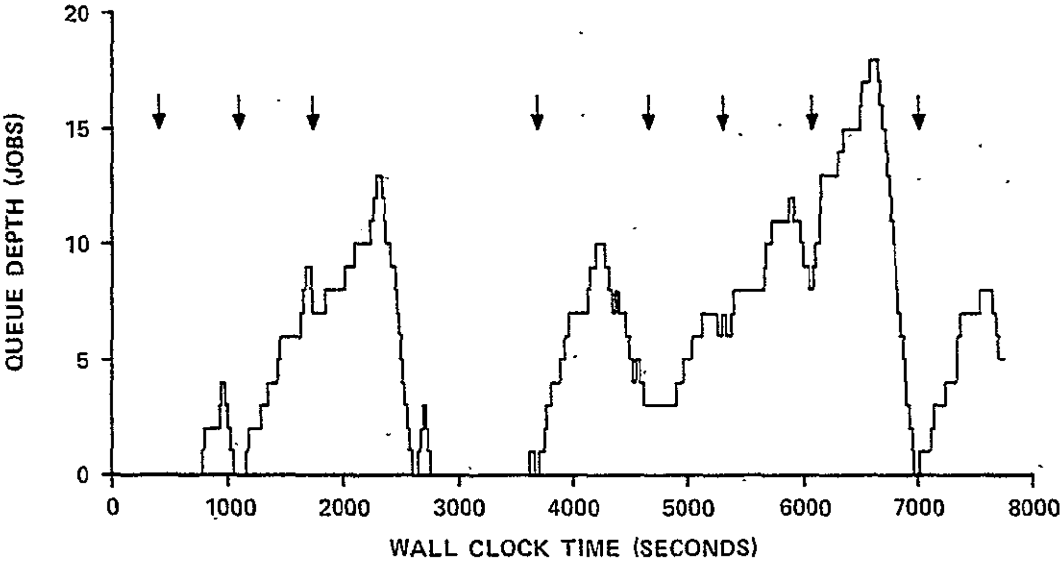


Figure 10. FMP Execution Queue. Arrows Mark the Beginning of FMP Execution on a Model 4 Job.

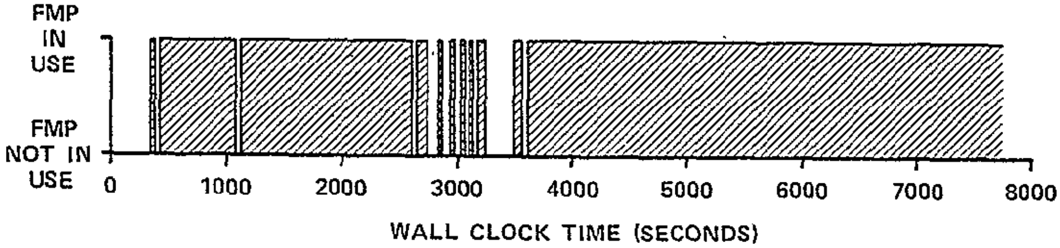


Figure 11. FMP Utilization

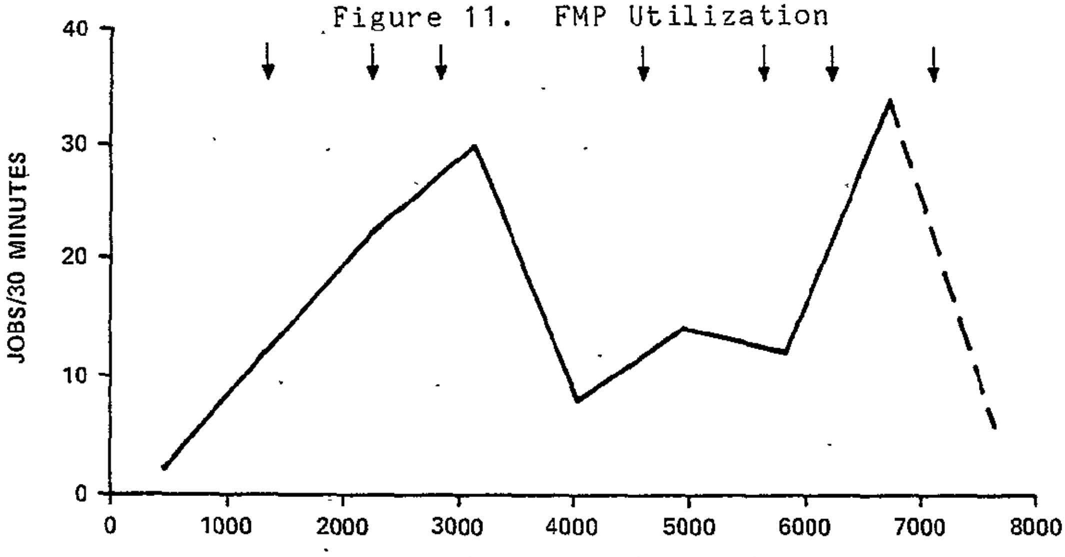


Figure 12. Job Throughput. Arrows Mark the Completion of Model 4 Jobs.

FMP, as shown in FMP queue, is very effectively running at full tilt. Throughput has slowed because three Model #4 jobs, 1800 FMP seconds, are being processed. This blocks out many jobs, as manifested by the growing queue. Near clock time 3000 and 7000 seconds notice how quickly the system finishes jobs after the FMP has stopped queuing them. This shows the rapid response of the communications network, DSK and SPS, and is very encouraging.

Arrows in figure 12 denote job completion times for the Model 4 jobs. Comparison with figure 7 shows the turnaround time for these jobs: 34.3 ± 8.0 minutes. This time includes 10 minutes of FMP processing and 13 minutes of SPS pre- and post-processing. Such an impressive turnaround statistic is dearly paid for by the turnaround rate of the other three job classes. Note table 1 below.

Table 1. Turnaround Statistics

Job Class	Turnaround time ± sample standard deviation	CPU time required (FMP and SPS)
Model 1	18.5 ± 8.7 min	51 ± 1 sec
Model 2	23.8 ± 10.8 min	3.9 ± 2 min
Model 3	29.7 ± 11.1 min	8.0 ± 2 min
Model 4	34.3 ± 8.0 min	23 ± 2 min

The final throughput mean for the run, 16.5 jobs/30 min (or 33 job/hr; see last page of listing in appendix A) is somewhat misleading for two rather coupled reasons. First, some ten jobs are well into the system and are partially completed. They are not included in this statistic, nor in the last point on the throughput graph. Second, a Model #4 job finished FMP execution at clock time 7533 seconds, 2 minutes before "simulation stop", and has left a building FMP execute queue behind. Thus, just as an FMP blocking condition was showing its bad side, i.e., diminished system throughput, the simulator stopped.

4.2.2 General System Response

During 2 hours of simulation forty-four Model 1, fifteen Model 2, four Model 3, and seven Model 4 jobs were begun and completed. An additional fourteen jobs had begun of which 10 were well on their way to completion. A total of approximately 1.5×10^{8} 64-bit words traveled within the LCN; in most cases they traveled twice - both to and from the DSK. Over 6×10^{12} floating-point operations were performed in the FMP and another 5×10^{10} floating-point operations within the SPS. These numbers, according to the Ames' Usage Model, represent a typical load during a two-hour period.

Aside from the FMP, no system hardware is backed up. The SPS mean utilizations are 47% for SPS1, and 45% for SPS2. GRF utilization is entirely due to workshop usage. The bit-serial data trunks are in no case more than 10% utilized. PDCs ranged from 2% to 9% usage. The majority of disk seeks are sector selects or minimum cylinder selects. In short, there were no traffic complications due to data transfers. Files of up to 10 million words were transferred in 14 seconds (e.g., Job #97; FMP --> DSK). The longest time for a file transfer was 100 seconds for 5 million words from the DSK to SPS (Job #96).

4.2.3 Variations on the Example

4.2.3.1 Different Arrival Statistics

Simulations were run with a different set of actual arrival times; all interarrival mean times were left unchanged. Though basic results were the same, one point was greatly clarified. The arrival profile of Model 4 jobs is the driving force in the system utilization. When these jobs are sparse, the FMP utilization drops to a value necessary to meet needs. For example, in one case an arrival rate for Model 4 jobs, 3.0 jobs/hr (compared to 4.0/hr in the above example), resulted in an FMP utilization of only 70%. In that case the FMP queue averaged only 2.3 jobs, and the system never had trouble keeping up - nor should it have.

4.2.3.2 Different LCN Geometry

Comparable, in some cases identical, workloads and arrival times were run through LCN1 and LCN3. Since the LCN is so lightly utilized, no appreciable difference on system response is expected (see light-load/heavy-load description in sections 3.3 and 3.4). LCN 3 showed no difference in general system utilization. Throughput turnaround, and processor utilization were virtually identical. Utilization of Trunks 1 and 2 doubled to 4% along with the utilization of PDCs 4 and 5. LCN1 showed slight degradation in throughput, and file transport times between the SPS and DSK. These transfer times were increased because Trunk 6 has to service both SPS1 and SPS2 paths to the DSK. Trunk 6 utilization went up to 17%. General system response was still very similar.

4.2.3.3 Effect of Change in SPS Performance and Load

The SPS processing load was arbitrarily decreased by 50%. This lead to an interesting result: The only differences in system response were:

1) SPS utilization diminshed by the appropriate amount,

2) turnaround times were slightly decreased.

Throughput, FMP utilization, and the FMP queue profile were essentially unchanged. This shows that the SPS setup and post-processing work is entirely hidden by the FMP workload. Support work done elsewhere in the NASF does not degrade the efficiency of the system, except during system startup. Thus, in this case, system efficiency is synonymous with FMP utilization.

SPS performance was then degraded by varying amounts: 2 devices of 3-megaflop/sec computation rates each, then computation rates of 2.5 megaflops/sec each and 2.0 megaflops/sec each. Though the demands on the SPS devices increased, the SPS did not saturate. For 2.0-megaflops/sec devices the utilization bulged to 71% for the lead SPS device and 66% for the backup. The average depth of the SPS execution queues were 1.1 and 0.7 jobs, respectively. The job throughput and FMP utilization were essentially unchanged. Average turnaround per job was longer due to the slower pre- and post-processing times, as expected. The interesting point is that no added system overhead was injected. The SPS work is still almost entirely hidden under the FMP work. As the SPS computational horsepower falls below 2 megaflops/sec this will no longer be true, for at that point saturation is about to set in.

4.2.3.4 Effect of Changing the Priority of a Job Class

One simulation was done with Model #4 jobs given lowest priority relative to the other prime-shift users. Two obvious results were noticed. Total job throughput went up by some 30% at the expense of Model #4 throughput which decreased by about 20%. Turnaround times for jobs of models #1, #2, and #3 were also improved significantly at the expense of Model #4 jobs. Demands on the SPS data trunks, PDCs, and disks were unchanged. But the FMP utilization, and thus the use of the system, decreased to 75% (down from 86%). Whether this trend is general or is a statistical fluctuation that would disappear with several runs is simply not clear at this time. The tradeoffs between the Model #4 jobs and the other jobs is clear, though more runs should yield more accurate performance tradeoffs. In any case, such hypotheses are easily tested via liberal use of the simulator.

4.2.3.5 Simulation of the Night Shift Workload

A two-hour simulation of the night shift (Usage Model job classes 5, 6, and 7) has also been done. All data transfers were modeled (e.g. 75 million-word restart files). Post-processing of the Model #6 and Model #7 "contour movies" was left out. A simple calculation shows that during a 10-hour night shift, the movie processing demand on the SPS is simply too great:

$$\frac{\text{movie flops}}{\text{night}} = \frac{7500 \text{ frames}}{\text{night}} \times \frac{10{,}000 \text{ pts}}{\text{frame}} \times \frac{6 \times 10^{7} \text{ flops}}{4{,}000 \text{ pts.}} \times \frac{\text{night}}{10 \text{ hr}}$$

$$= 1.125 \times 10^{12} \text{ flops/10 hr or } 31.3 \text{ megaflops/sec.}$$

Obviously this user demand causes concern for it outbalances any other SPS demand stated by the Usage Model by at least an order of magnitude. A system that meets this demand outside of the FMP will have excessive horsepower for the remainder of the day. And it is not clear at this time that such a task does not belong in the FMP.

With this one point aside, the system had no problem with the traffic flow problem.

FMP Utilization	87%	
SPS1 Utilization	12%\	No workshop usage.
SPS2 Utilization	2%/	
Trunk 6 Utilization	11%	
All other trunks	1%	

In other words the system was wide open and the FMP was well stocked with work.

4.3 Conclusions

This family of examples addresses only the first pass of system evaluation. The results pose new questions. For example at what cost can turnaround times for the more numerous "simple" problems be shortened? Can throughput be more consistent in time? How heavily utilized can the SPSs be before system efficiency is significantly diminished? Clearly, the system designers could simulate for years playing with job priority algorithms, workload arrival scenarios, keeping the number of jobs executed unchanged. At some juncture the tradeoffs demanded by the needs of different job classes will be assessed, invariably yielding some new questions and tradeoffs.

Some truths have come from this first pass simulation which are maxims to the Ames concept of the task to be done and the machine that is to do them.

1) The NASF as proposed in this paper has no system bottlenecks other than the FMP. Job functions peripheral to the FMP processing are entirely hidden by the FMP operations so long as the workload is sufficiently heavy to keep it busy. Thus, for the present Usage Model, system efficiency is the FMP utilization.

2) Twenty hours of FMP computations cannot, in practice, be done in twenty hours. Aside from system start-up and wind-down time, aside from breakdown possibilities, if the FMP ever goes idle during the running day that time is lost, never to be recaptured. According to the present Usage Model a certain random nature in job arrivals

   seems evident. This simply implies a high probability that at some time(s) during the day the job arrival profile and the load within the system will not keep the FMP busy.

3) To obtain high throughput the FMP must be highly utilized. In practicality this requires substantial queuing activity for the FMP, i.e., a healthy FMP execute queue. Thus for a system workload which demands high FMP utilization, there is a tradeoff between throughput and job turnaround. Simulation shows that the greater the extent to which the FMP queue is kept non-empty, the greater the system throughput. Such queuing clearly cuts down on job turnaround.

   A particular class of jobs may occupy the queue for relatively long periods of time (a good possibility) or the general user community (average) job may occupy it for a somewhat shorter period. Unfortunately, high throughput for one class of jobs may create long turnaround for jobs of another class. Conversely, large volume of short turnaround jobs decrease FMP utlization and throughput.

This simulation has also identified some possible system difficulties. Though the system concept seems very encouraging, playing the devil's advocate is always fun: two thorns are foreseeable. First, results to date are based on a very sketchy understanding of the capabilities of the SPS system and the load it must carry. Assumptions and guesses have been clearly stated in sections 2, 3, and 4, but it is conceivable that they are not better than 50% accurate and perhaps worse. Second, what if one SPS device should break down? This is certain to occur on occasion. According to present simulation runs, one SPS could not handle the entire load. It could, however, do a reasonable job if the workload were diminished somewhat. Simulation of such a situation remains to be done.

Appendix A

SYSTEM SIMULATOR OUTPUT EXAMPLE

```
$$$$$$$$    $$$$$$    $$$$$$$$   $      $   $$    $$   $$$$$$$$   $      $
   $$      $$    $$      $$      $$    $$   $$    $$   $$         $$    $$
   $$      $$            $$      $$$  $$$   $$    $$   $$         $$$  $$$
   $$       $$$$$$       $$      $$$$$$$$   $$    $$   $$$$$$$    $$$$$$$$
   $$            $$      $$      $$ $$ $$   $$    $$         $$   $$ $$ $$
   $$            $$      $$      $$    $$   $$    $$         $$   $$    $$
   $$            $$      $$      $$    $$   $$    $$         $$   $$    $$
   $$      $$    $$      $$      $$    $$   $$    $$   $$    $$   $$    $$
$$$$$$$$    $$$$$$    $$$$$$$$   $$    $$    $$$$$$     $$$$$$    $$    $$

SUMMARY OF SYSTEM CONFIGURATION INPUT DATA

 1.  01,14,1,23,1,6,1                  FE1/DISK1
 2.  01,13,2,23,2,7,2
 3.  01,13,1,22,2,5,3
 4.  02,14,1,25,1,6,1                  FE1/DISK 2
 5.  02,13,2,25,2,7,2
 6.  02,13,1,24,2,5,3
 7.  03,14,1,27,1,6,1                  FE1/DISK 3
 8.  03,13,2,27,2,7,2
 9.  03,13,1,26,2,5,3
10.  04,14,1,29,1,6,1                  FE1/DISK 4
11.  04,13,2,29,2,7,2
12.  04,13,1,28,2,6,3
13.  01,18,1,23,2,7,1                  FE2/DISK 1
14.  01,17,2,23,1,6,2
15.  01,17,1,22,2,5,3
16.  02,18,1,25,2,7,1                  FE2/DISK 2
17.  02,17,2,25,1,6,2
18.  02,17,1,24,2,5,3
19.  03,18,1,27,2,7,1                  FE2/DISK 3
20.  03,17,2,27,1,6,2
21.  03,17,1,26,2,5,3
22.  04,18,1,29,2,7,1                  FE2/DISK 4
23.  04,17,2,29,1,6,2
24.  04,17,1,28,2,5,3
25.  11,04,1,22,1,1,1                  FMP/DISK 1
26.  11,02,1,22,2,5,2
27.  11,02,2,23,2,7,3
28.  12,05,1,24,1,2,1                  FMP/DISK 2
29.  12,02,1,24,2,5,2
30.  12,02,2,25,2,7,3
31.  13,06,1,26,1,3,1                  FMP/DISK 3
32.  13,02,1,26,2,5,2
33.  13,02,2,27,2,7,3
34.  14,07,1,28,1,4,1                  FMP/DISK 4
35.  14,02,1,28,2,5,2
36.  14,02,2,29,2,7,3
37.  10,13,1,01,1,5,1                  FE1/FMP
38.  10,13,1,02,1,5,2
39.  10,14,1,01,2,6,3
40.  10,17,1,01,1,5,1                  FE2/FMP
41.  10,17,1,02,1,5,2
42.  10,18,1,02,2,7,3
43.  21,37,2,23,2,7,1                  GRF1/DISK1
44.  21,37,1,22,2,5,2
45.  22,37,2,25,2,7,1                  GRF1/DISK2
46.  22,37,1,24,2,5,2
47.  23,37,2,27,2,7,1                  GRF1/DISK3
48.  23,37,1,26,2,5,2
49.  24,37,2,29,2,7,1                 GRF/DISK4
```

```
50.  24,37,1,28,2,5,2
51.  21,38,2,23,1,6,1              GRF2/DISK1
52.  21,38,1,22,2,5,2
53.  22,38,2,25,1,6,1
54.  22,38,1,24,2,5,2
55.  23,38,2,27,1,6,1
56.  23,38,1,26,2,5,2
57.  24,38,2,29,1,6,1
58.  24,38,1,28,2,5,2
59.  20,13,1,37,1,5,1               FE1/GRF1
60.  20,13,2,37,2,7,2
61.  20,17,1,37,1,5,1             FE2/GRF1
62.  20,18,1,37,2,7,3
63.  20,13,1,38,1,5,1               FE1/GRF2
64.  20,14,1,38,2,6,3
65.  20,17,1,38,1,5,1             FE2/GRF2
66.  20,17,2,38,2,6,2
67.  0,0,0,0,0,00,0
```

SUMMARY OF MESSAGE TRAFFIC INPUT DATA

```
 1.  10,0,120,2,0,0             MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000100
 2.  10,61,1,15,0,-600          1SEC ABORTED SPS COMPIL'N - WAIT 10 MIN. 000110
 3.  10,61,1,15,0,-1            SPS COMPILATION OF SOURCE CODE-1 SEC.   000120
 4.  11,1,0,0,1,-1              FILE1-LOAD MODULE 15K: SPS <--> DSK.    000130
 5.  11,10,0,1,1,0              FILE1 DSK <--> FMP.                     000140
 6.  12,21,0,0,1,-1             FILE2-CONFIGATION 10K: GRF <--> DSK.    000150
 7.  12,10,0,1,1,-1             FILE2 DSK <--> FMP.                     000160
 8.  10,60,10,0,2,-1            FMP FLOW CODE PROCESSING - 10 SEC.      000170
 9.  18,10,0,0,2,-1             FILE8-RESULTS FILE 60K: FMP <--> DSK.   000180
10.  18,1,0,1,2,-1              FILE8 DSK <--> SPS.                     000190
11.  10,61,60,12,1,-1           SPS POST-PROCESSING:40 SEC-25% LOAD.    000200
12.  10,0,0,0,0,0               JOB RUN COMPLETE.                       000210
13.  20,0,120,1,0,0             MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000220
14.  20,61,1,15,0,-600          1SEC ABORTED SPS COMPIL'N - WAIT 10 MIN. 000230
15.  20,61,1,15,0,-300          ABORTED COMPILATION AGAIN- WAIT 5 MIN.  000240
16.  20,61,1,15,0,-1            SPS COMPILATION OF SOURCE CODE-1 SEC.   000250
17.  21,1,0,0,1,-1              FILE1-LOAD MODULE 15K: SPS <--> DSK.    000260
18.  21,10,0,1,1,0              FILE1 DSK <--> FMP.                     000270
19.  22,21,0,0,1,-1             FILE2-CONFIGATION 10K: GRF <--> DSK.    000280
20.  22,10,0,1,1,-1             FILE2 DSK <--> FMP.                     000290
21.  20,60,10,0,2,-1            FMP FLOW CODE PROCESSING - 10 SEC.      000300
22.  28,10,0,0,2,-1             FILE8-RESULTS FILE 60K: FMP <--> DSK.   000310
23.  28,1,0,1,2,-1              FILE8 DSK <--> SPS.                     000320
24.  20,61,60,12,1,-1           SPS POST-PROCESSING:40 SEC-25% LOAD.    000330
25.  20,0,0,0,0,0               JOB RUN COMPLETE.                       000340
26.  30,0,120,1,0,0             MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000350
27.  30,61,1,15,0,-1            SPS COMPILATION OF SOURCE CODE-1 SEC.   000360
28.  31,1,0,0,1,-1              FILE1-LOAD MODULE 15K: SPS <--> DSK.    000370
29.  31,10,0,1,1,0              FILE1 DSK <--> FMP.                     000380
30.  32,21,0,0,1,-1             FILE2-CONFIGATION 10K: GRF <--> DSK.    000390
31.  32,10,0,1,1,-1             FILE2 DSK <--> FMP.                     000400
32.  30,60,10,0,2,-1            FMP FLOW CODE PROCESSING - 10 SEC.      000410
33.  38,10,0,0,2,-1             FILE8-RESULTS FILE 60K: FMP <--> DSK.   000420
34.  38,1,0,1,2,-1              FILE8 DSK <--> SPS.                     000430
35.  30,61,60,12,1,-1           SPS POST-PROCESSING:40 SEC-25% LOAD.    000440
36.  30,0,0,0,0,0               JOB RUN COMPLETE.                       000450
37.  40,0,120,1,0,0             MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000460
38.  40,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 MIN.000470
```

```
 39.  40,61,1,15,0,-1     SPS COMPILATION OF SOURCE CODE-1 SEC.    000480
 40.  41,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     000490
 41.  41,10,0,1,1,0       FILE1 DSK <--> FMP.                      000500
 42.  42,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     000510
 43.  42,10,0,1,1,-1      FILE2 DSK <--> FMP.                      000520
 44.  40,60,10,0,2,-1     FMP FLOW CODE PROCESSING - 10 SEC.       000530
 45.  49,10,0,0,90,-1      FILE9-DEBUG DUMP 3M; FMP <--> DSK.      000540
 46.  49,1,0,1,90,-1       FILE9 DSK <--> SPS.                     000550
 47.  40,61,60,12,1,-1    SPS POST-PROCESSING;40 SEC-25% LOAD.     000560
 48.  40,0,0,0,0,0        JOB RUN COMPLETE.                        000570
 49.  50,0,120,1,0,0      MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000580
 50.  50,61,1,15,0,-600     1 SEC ABORTED SPS COMPIL'N - WAIT 10 MI000590
 51.  50,61,1,15,0,-1     SPS RECOMPILATION OF SOURCE CODE-1 SEC.  000600
 52.  51,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     000610
 53.  51,10,0,1,1,0       FILE1 DSK <--> FMP.                      000620
 54.  52,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     000630
 55.  52,10,0,1,1,-1      FILE2 DSK <--> FMP.                      000640
 56.  50,60,10,0,2,-1     FMP FLOW CODE PROCESSING - 10 SEC.       000650
 57.  58,10,0,0,2,-1      FILE8-RESULTS FILE 60K; FMP <--> DSK.    000660
 58.  58,1,0,1,2,-1       FILE8 DSK <--> SPS.                      000670
 59.  50,61,60,12,1,-1    SPS POST-PROCESSING;40 SEC-25% LOAD.     000680
 60.  50,0,0,0,0,0        JOB RUN COMPLETE.                        000690
 61.  60,0,120,1,0,0      MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000700
 62.  60,61,1,15,0,-600     1 SEC ABORTED SPS COMPIL'N - WAIT 10 MI000710
 63.  60,61,1,15,0,-300     ABORTED COMPIL'N AGAIN - WAIT 5 MIN. TH000720
 64.  60,61,1,15,0,-1     SPS RECOMPILATION OF SOURCE CODE-1 SEC.  000730
 65.  61,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     000740
 66.  61,10,0,1,1,0       FILE1 DSK <--> FMP.                      000750
 67.  62,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     000760
 68.  62,10,0,1,1,-1      FILE2 DSK <--> FMP.                      000770
 69.  60,60,10,0,2,-1     FMP FLOW CODE PROCESSING - 10 SEC.       000780
 70.  68,10,0,0,2,-1      FILE8-RESULTS FILE 60K; FMP <--> DSK.    000790
 71.  68,1,0,1,2,-1       FILE8 DSK <--> SPS.                      000800
 72.  60,61,60,12,1,-1    SPS POST-PROCESSING;40 SEC-25% LOAD.     000810
 73.  60,0,0,0,0,0        JOB RUN COMPLETE.                        000820
 74.  70,0,120,1,0,0      MODEL 1-INTERARRIVAL TIME NEAN OF 120 SEC000830
 75.  70,61,1,15,0,-1     SPS COMPILATION OF SOURCE CODE-1 SEC.    000840
 76.  71,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     000850
 77.  71,10,0,1,1,0       FILE1 DSK <--> FMP.                      000860
 78.  72,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     000870
 79.  72,10,0,1,1,-1      FILE2 DSK <--> FMP.                      000880
 80.  70,60,10,0,2,-1     FMP FLOW CODE PROCESSING - 10 SEC.       000890
 81.  78,10,0,0,2,-1      FILE8-RESULTS FILE 60K; FMP <--> DSK.    000900
 82.  78,1,0,1,2,-1       FILE8 DSK <--> SPS.                      000910
 83.  70,61,60,12,1,-1    SPS POST-PROCESSING;40 SEC-25% LOAD.     000920
 84.  70,0,0,0,0,0        JOB RUN COMPLETE.                        000930
 85.  80,0,120,1,0,0      MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC000940
 86.  80,61,1,15,0,-600     1 SEC ABORTED SPS COMPIL'N - WAIT 10 MI000950
 87.  80,61,1,15,0,-1     SPS RECOMPILATION OF SOURCE CODF-1 SEC.  000960
 88.  81,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     000970
 89.  81,10,0,1,1,0       FILE1 DSK <--> FMP.                      000980
 90.  82,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     000990
 91.  82,10,0,1,1,-1      FILE2 DSK <--> FMP.                      001000
 92.  80,60,10,0,2,-1     FMP FLOW CODE PROCESSING - 10 SEC.       001010
 93.  89,10,0,0,90,-1      FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK. 001020
 94.  89,1,0,1,90,-1       FILE9 DSK <--> SPS.                     001030
 95.  80,61,60,12,1,-1    SPS POST-PROCESSING;40 SEC-25% LOAD.     001040
 96.  80,0,0,0,0,0        JOB RUN COMPLETE.                        001050
 97.  90,0,120,1,0,0      MODEL 1-INTERARRIVAL TIME MEAN OF 120 SEC001060
 98.  90,61,1,15,0,-600     1 SEC ABORTED SPS COMPIL'N - WAIT 10 MI001070
 99.  90,61,1,15,0,-1     SPS RECOMPILATION OF SOURCE CODE-1 SEC.  001080
100.  91,1,0,0,1,-1       FILE1-LOAD MODULE 15K; SPS <--> DSK.     001090
101.  91,10,0,1,1,0       FILE1 DSK <--> FMP.                      001100
102.  92,21,0,0,1,-1      FILE2-CONFIGATION 10K; GRF <--> DSK.     001110
```

```
103.  92,10,0,1,1,-1                   FILE2 DSK <--> FMP,                      001120
104.  90,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.       001130
105.  98,10,0,0,2,-1                   FILE8-RESULTS FILE 60K; FMP <--> DSK.    001140
106.  98,1,0,1,2,-1                    FILE8 DSK <--> SPS.                      001150
107.  90,61,60,12,1,-1                 SPS POST-PROCESSING;40 SEC-25% LOAD.     001160
108.  90,0,0,0,0,0                     JOB RUN COMPLETE.                        001170
109.  100,0,120,1,0,0                   MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001180
110.  100,61,1,15,0,-600                   1 SEC ABORTED SPS COMPIL'N - WAIT 10 M001190
111.  100,61,1,15,0,-300                   ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T001200
112.  100,61,1,15,0,-1                  SPS RECOMPILATION OF SOURCE CODE-1 SEC. 001210
113.  101,1,0,0,1,-1                    FILE1-LOAD MODULE 15K; SPS <--> DSK     001220
114.  101,10,0,1,1,0                    FILE1 DSK <--> FMP.                     001230
115.  102,21,0,0,1,-1                   FILE2-CONFIGATION 10K; GRF <--> DSK.    001240
116.  102,10,0,1,1,-1                   FILE2 DSK <--> FMP.                     001250
117.  100,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.      001260
118.  108,10,0,0,2,-1                   FILE8-RESULTS FILE 60K; FMP <--> DSK.   001270
119.  108,1,0,1,2,-1                    FILE8 DSK <--> SPS.                     001280
120.  100,61,60,12,1,-1                 SPS POST-PROCESSING;40 SEC-25% LOAD.    001290
121.  100,0,0,0,0,0                     JOB RUN COMPLETE.                       001300
122.  110,0,120,1,0,0                   MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001310
123.  110,61,1,15,0,-1                  SPS COMPILATION OF SOURCE CODE-1 SEC.   001320
124.  111,1,0,0,1,-1                    FILE1-LOAD MODULE 15K; SPS <--> DSK.    001330
125.  111,10,0,1,1,0                    FILE1 DSK <--> FMP.                     001340
126.  112,21,0,0,1,-1                   FILE2-CONFIGATION 10K; GRF <--> DSK.    001350
127.  112,10,0,1,1,-1                   FILE2 DSK <--> FMP.                     001360
128.  110,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.      001370
129.  119,10,0,0,90,-1                   FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.001380
130.  119,1,0,1,90,-1                    FILE9 DSK <--> SPS.                    001390
131.  110,61,60,12,1,-1                 SPS POST-PROCESSING;40 SEC-25% LOAD.    001400
132.  110,0,0,0,0,0                     JOB RUN COMPLETE.                       001410
133.  120,0,120,1,0,0                   MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001420
134.  120,61,1,15,0,-600                  1 SEC ABORTED SPS COMPIL'N - WAIT 10 M001430
135.  120,61,1,15,0,-1                  SPS RECOMPILATION OF SOURCE CODE-1 SEC. 001440
136.  121,1,0,0,1,-1                    FILE1-LOAD MODULE 15K; SPS <--> DSK     001450
137.  121,10,0,1,1,0                    FILE1 DSK <--> FMP.                     001460
138.  122,21,0,0,1,-1                   FILE2-CONFIGATION 10K; GRF <--> DSK.    001470
139.  122,10,0,1,1,-1                   FILE2 DSK <--> FMP.                     001480
140.  120,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.      001490
141.  129,10,0,0,90,-1                   FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.001500
142.  129,1,0,1,90,-1                    FILE9 DSK <--> SPS..                   001510
143.  120,61,60,12,1,-1                 SPS POST-PROCESSING;40 SEC-25% LOAD.    001520
144.  120,0,0,0,0,0                     JOB RUN COMPLETE.                       001530
145.  130,0,120,1,0,0                   MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001540
146.  130,61,1,15,0,-600                  1 SEC ABORTED SPS COMPIL'N - WAIT 10 M001550
147.  130,61,1,15,0,-1                  SPS RECOMPILATION OF SOURCE CODE-1 SEC. 001560
148.  131,1,0,0,1,-1                    FILE1-LOAD MODULE 15K; SPS <--> DSK.    001570
149.  131,10,0,1,1,0                    FILE1 DSK <--> FMP.                     001580
150.  132,21,0,0,1,-1                   FILE2-CONFIGATION 10K; GRF <--> DSK.    001590
151.  132,10,0,1,1,-1                   FILE2 DSK <--> FMP.                     001600
152.  130,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.      001610
153.  138,10,0,0,2,-1                   FILE8-RESULTS FILE 60K; FMP <--> DSK.   001620
154.  138,1,0,1,2,-1                    FILE8 DSK <--> SPS.                     001630
155.  130,61,60,12,1,-1                 SPS POST-PROCESSING;40 SEC-25% LOAD.    001640
156.  130,0,0,0,0,0                     JOB RUN COMPLETE.                       001650
157.  140,0,120,1,0,0                   MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001660
158.  140,61,1,15,0,-600                  1 SEC ABORTED SPS COMPIL'N - WAIT 10 M001670
159.  140,61,1,15,0,-300                  ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T001680
160.  140,61,1,15,0,-1                  SPS RECOMPILATION OF SOURCE CODE-1 SEC. 001690
161.  141,1,0,0,1,-1                    FILE1-LOAD MODULE 15K; SPS <--> DSK.    001700
162.  141,10,0,1,1,0                    FILE1 DSK <--> FMP.                     001710
163.  142,21,0,0,1,-1                   FILE2-CONFIGATION 10K; GRF <--> DSK.    001720
164.  142,10,0,1,1,-1                   FILE2 DSK <--> FMP.                     001730
165.  140,60,10,0,2,-1                  FMP FLOW CODE PROCESSING - 10 SEC.      001740
```

```
166.  148,10,0,0,2,-1              FILE8-RESULTS FILE 60K; FMP <--> DSK.    001750
167.  148,1,0,1,2,-1               FILE8 DSK <--> SPS.                      001760
168.  140,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     001770
169.  140,0,0,0,0,0                JOB RUN COMPLETE.                        001780
170.  150,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001790
171.  150,61,1,15,0,-1             SPS COMPILATION OF SOURCE CODE-1 SEC.    001800
172.  151,1,0,0,1,-1               FILE1-LOAD MODULE 15K; SPS <--> DSK.     001810
173.  151,10,0,1,1,0               FILE1 DSK <--> FMP.                      001820
174.  152,21,0,0,1,-1              FILE2-CONFIGATION 10K; GRF <--> DSK.     001830
175.  152,10,0,1,1,-1              FILE2 DSK <--> FMP.                      001840
176.  150,60,10,0,2,-1             FMP FLOW CODE PROCESSING - 10 SEC.       001850
177.  158,10,0,0,2,-1              FILE8-RESULTS FILE 60K; FMP <--> DSK.    001860
178.  158,1,0,1,2,-1               FILE8 DSK <--> SPS.                      001870
179.  150,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     001880
180.  150,0,0,0,0,0                JOB RUN COMPLETE.                        001890
181.  160,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE001900
182.  160,61,1,15,0,-600             1 SEC ABORTED SPS COMPIL'N - WAIT 10 M001910
183.  160,61,1,15,0,-1             SPS RECOMPILATION OF SOURCE CODE-1 SEC. 001920
184.  161,1,0,0,1,-1               FILE1-LOAD MODULE 15K; SPS <--> DSK.     001930
185.  161,10,0,1,1,0               FILE1 DSK <--> FMP.                      001940
186.  162,21,0,0,1,-1              FILE2-CONFIGATION 10K; GRF <--> DSK.     001950
187.  162,10,0,1,1,-1              FILE2 DSK <--> FMP.                      001960
188.  160,60,10,0,2,-1             FMP FLOW CODE PROCESSING - 10 SEC.       001970
189.  169,10,0,0,90,-1              FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.001980
190.  169,1,0,1,90,-1               FILE9 DSK <--> SPS.                     001990
191.  160,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     002000
192.  160,0,0,0,0,0                JOB RUN COMPLETE.                        002010
193.  170,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002020
194.  170,61,1,15,0,-600             1 SEC ABORTED SPS COMPIL'N - WAIT 10 M002030
195.  170,61,1,15,0,-1             SPS RECOMPILATION OF SOURCE CODE-1 SEC. 002040
196.  171,1,0,0,1,-1               FILE1-LOAD MODULE 15K; SPS <--> DSK.     002050
197.  171,10,0,1,1,0               FILE1 DSK <--> FMP.                      002060
198.  172,21,0,0,1,-1              FILE2-CONFIGATION 10K; GRF <--> DSK.     002070
199.  172,10,0,1,1,-1              FILE2 DSK <--> FMP.                      002080
200.  170,60,10,0,2,-1             FMP FLOW CODE PROCESSING - 10 SEC.       002090
201.  178,10,0,0,2,-1              FILE8-RESULTS FILE 60K; FMP <--> DSK.    002100
202.  178,1,0,1,2,-1               FILE8 DSK <--> SPS.                      002110
203.  170,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     002120
204.  170,0,0,0,0,0                JOB RUN COMPLETE.                        002130
205.  180,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002140
206.  180,61,1,15,0,-600             1 SEC ABORTED SPS COMPIL'N - WAIT 10 M002150
207.  180,61,1,15,0,-300             ABORTED COMPIL'N AGAIN - WAIT 5 MIN. T002160
208.  180,61,1,15,0,-1             SPS RECOMPILATION OF SOURCE CODE-1 SEC. 002170
209.  181,1,0,0,1,-1               FILE1-LOAD MODULE 15K; SPS <--> DSK.     002180
210.  181,10,0,1,1,0               FILE1 DSK <--> FMP.                      002190
211.  182,21,0,0,1,-1              FILE2-CONFIGATION 10K; GRF <--> DSK.     002200
212.  182,10,0,1,1,-1              FILE2 DSK <--> FMP.                      002210
213.  180,60,10,0,2,-1             FMP FLOW CODE PROCESSING - 10 SEC.       002220
214.  188,10,0,0,2,-1              FILE8-RESULTS FILE 60K; FMP <--> DSK.    002230
215.  188,1,0,1,2,-1               FILE8 DSK <--> SPS.                      002240
216.  180,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     002250
217.  180,0,0,0,0,0                JOB RUN COMPLETE.                        002260
218.  190,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002270
219.  190,61,1,15,0,-1             SPS COMPILATION OF SOURCE CODE-1 SEC.    002280
220.  191,1,0,0,1,-1               FILE1-LOAD MODULE 15K; SPS <--> DSK.     002290
221.  191,10,0,1,1,0               FILE1 DSK <--> FMP.                      002300
222.  192,21,0,0,1,-1              FILE2-CONFIGATION 10K; GRF <--> DSK.     002310
223.  192,10,0,1,1,-1              FILE2 DSK <--> FMP.                      002320
224.  190,60,10,0,2,-1             FMP FLOW CODE PROCESSING - 10 SEC.       002330
225.  199,10,0,0,90,-1              FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.002340
226.  199,1,0,1,90,-1               FILE9 DSK <--> SPS.                     002350
227.  190,61,60,12,1,-1            SPS POST-PROCESSING;40 SEC-25% LOAD.     002360
228.  190,0,0,0,0,0                JOB RUN COMPLETE.                        002370
229.  200,0,120,1,0,0              MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002380
```

```
200,61,1,15,0,-600                   1 SEC ABORTED SPS COMPIL'N - WAIT 10 M002390
200,61,1,15,0,-1                   SPS RECOMPILATION OF SOURCE CODE-1 SEC. 002400
201,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002410
201,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002420
202,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002430
202,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     002440
200,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      002450
209,10,0,0,90,-1                    FILE9-DEBUG DUMP FILE.3M; FMP <--> DSK.002460
209,1,0,1,90,-1                     FILE9 DSK <--> SPS.                    002470
200,61,60,12,1,-1                  SPS POST-PROCESSING;40 SEC-25% LOAD.    002480
200,0,0,0,0,0                      JOB RUN COMPLETE.                       002490
210,0,120,1,0,0                    MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002500
210,61,1,15,0,-1                   SPS COMPILATION OF SOURCE CODE-1 SEC.   002510
211,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002520
211,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002530
212,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002540
212,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     002550
210,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      002560
218,10,0,0,2,-1                    FILE8-RESULTS FILE 60K; FMP <--> DSK.   002570
218,1,0,1,2,-1                     FILE8 DSK <--> SPS.                     002580
210,61,60,12,1,-1                  SPS POST-PROCESSING;40 SEC-25% LOAD.    002590
210,0,0,0,0,0                      JOB RUN COMPLETE.                       002600
220,0,120,1,0,0                    MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002610
220,61,1,15,0,-1                   SPS COMPILATION OF SOURCE CODE-1 SEC.   002620
221,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002630
221,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002640
222,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002650
222,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     002660
220,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      002670
228,10,0,0,2,-1                    FILE8-RESULTS FILE 60K; FMP <--> DSK.   002680
228,1,0,1,2,-1                     FILE8 DSK <--> SPS.                     002690
220,61,60,12,1,-1                  SPS POST-PROCESSING;40 SEC-25% LOAD.    002700
220,0,0,0,0,0                      JOB RUN COMPLETE.                       002710
230,0,120,1,0,0                    MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002720
230,61,1,15,0,-1                   SPS COMPILATION OF SOURCE CODE-1 SEC.   002730
231,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002740
231,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002750
232,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002760
232,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     002770
230,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      002780
238,10,0,0,2,-1                    FILE8-RESULTS FILE 60K; FMP <--> DSK.   002790
238,1,0,1,2,-1                     FILE8 DSK <--> SPS.                     002800
230,61,60,12,1,-1                  SPS POST-PROCESSING;40 SEC-25% LOAD.    002810
230,0,0,0,0,0                      JOB RUN COMPLETE.                       002820
240,0,120,1,0,0                    MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002830
240,61,1,15,0,-1                   SPS COMPILATION OF SOURCE CODE-1 SEC.   002840
241,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002850
241,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002860
242,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002870
242,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     002880
240,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      002890
249,10,0,0,90,-1                    FILE9-DEBUG DUMP 3M; FMP <--> DSK.     002900
249,1,0,1,90,-1                     FILE9 DSK <--> SPS.                    002910
240,61,60,12,1,-1                  SPS POST-PROCESSING;40 SEC-25% LOAD.    002920
240,0,0,0,0,0                      JOB RUN COMPLETE.                       002930
250,0,120,1,0,0                    MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE002940
250,61,1,15,0,-600                   1 SEC ABORTED SPS COMPIL'N - WAIT 10 M002950
250,61,1,15,0,-1                   SPS RECOMPILATION OF SOURCE CODE-1 SEC. 002960
251,1,0,0,1,-1                     FILE1-LOAD MODULE 15K; SPS <--> DSK.    002970
251,10,0,1,1,0                     FILE1 DSK <--> FMP.                     002980
252,21,0,0,1,-1                    FILE2-CONFIGATION 10K; GRF <--> DSK.    002990
252,10,0,1,1,-1                    FILE2 DSK <--> FMP.                     003000
250,60,10,0,2,-1                   FMP FLOW CODE PROCESSING - 10 SEC.      003010
258,10,0,0,2,-1                    FILE8-RESULTS FILE 60K; FMP <--> DSK.   003020
```

```
294.  258,1,0,1,2,-1                  FILE8 DSK <--> SPS.                          003030
295.  250,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003040
296.  250,0,0,0,0,0                   JOB RUN COMPLETE.                            003050
297.  260,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003060
298.  260,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003070
299.  260,61,1,15,0,-300                ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T003080
300.  260,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEC. 003090
301.  261,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.         003100
302.  261,10,0,1,1,0                  FILE1 DSK <--> FMP.                          003110
303.  262,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.         003120
304.  262,10,0,1,1,-1                 FILE2 DSK <--> FMP.                          003130
305.  260,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.           003140
306.  268,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.        003150
307.  268,1,0,1,2,-1                  FILE8 DSK <--> SPS.                          003160
308.  260,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003170
309.  260,0,0,0,0,0                   JOB RUN COMPLETE.                            003180
310.  270,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 14         003190
311.  270,61,1,15,0,-1                SPS COMPILATION OF SOURCE CODE-1 SEC         003200
312.  271,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DS           003210
313.  271,10,0,1,1,0                  FILE1 DSK <--> FMP.                          003220
314.  272,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.         003230
315.  272,10,0,1,1,-1                 FILE2 DSK <--> FMP.                          003240
316.  270,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.           003250
317.  278,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.        003260
318.  278,1,0,1,2,-1                  FILE8 DSK <--> SPS.                          003270
319.  270,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003280
320.  270,0,0,0,0,0                   JOB RUN COMPLETE.                            003290
321.  280,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003300
322.  280,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003310
323.  280,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEr. 003320
324.  281,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.         003330
325.  281,10,0,1,1,0                  FILE1 DSK <--> FMP.                          003340
326.  282,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.         003350
327.  282,10,0,1,1,-1                 FILE2 DSK <--> FMP.                          003360
328.  280,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.           003370
329.  289,10,0,0,90,-1                 FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.003380
330.  289,1,0,1,90,-1                  FILE9 DSK <--> SPS.                         003390
331.  280,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003400
332.  280,0,0,0,0,0                   JOB RUN COMPLETE.                            003410
333.  290,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003420
334.  290,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003430
335.  290,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEr. 003440
336.  291,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.         003450
337.  291,10,0,1,1,0                  FILE1 DSK <--> FMP.                          003460
338.  292,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.         003470
339.  292,10,0,1,1,-1                 FILE2 DSK <--> FMP.                          003480
340.  290,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.           003490
341.  298,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.        003500
342.  298,1,0,1,2,-1                  FILE8 DSK <--> SPS.                          003510
343.  290,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003520
344.  290,0,0,0,0,0                   JOB RUN COMPLETE.                            003530
345.  300,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003540
346.  300,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003550
347.  300,61,1,15,0,-300                ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T003560
348.  300,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEC. 003570
349.  301,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.         003580
350.  301,10,0,1,1,0                  FILE1 DSK <--> FMP.                          003590
351.  302,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.         003600
352.  302,10,0,1,1,-1                 FILE2 DSK <--> FMP.                          003610
353.  300,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.           003620
354.  309,10,0,0,90,-1                 FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.003630
355.  309,1,0,1,90,-1                  FILE9 DSK <--> SPS.                         003640
356.  300,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.         003650
357.  300,0,0,0,0,0                   JOB RUN COMPLETE.                            003660
```

```
358.  310,0,120,1,0,0           MODFL 1-INTERARRIVAL TIME MEAN OF 120 SE003670
359.  310,61,1,15,0,-1          SPS COMPILATION OF SOURCE CODE-1 SEC.   003680
360.  311,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    003690
361.  311,10,0,1,1,0            FILE1 DSK <--> FMP.                     003700
362.  312,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    003710
363.  312,10,0,1,1,-1           FILE2 DSK <--> FMP.                     003720
364.  310,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      003730
365.  318,10,0,0,2,-1           FILE8-RESULTS FILE 40K; FMP <--> DSK.   003740
366.  318,1,0,1,2,-1            FILFB DSK <--> SPS.                     003750
367.  310,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    003760
368.  310,0,0,0,0,0             JOB RUN COMPLETE.                       003770
369.  320,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003780
370.  320,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003790
371.  320,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 003800
372.  321,1,0,0,1,-1            FILFI-LOAD MODULE 15K; SPS <--> DSK.    003810
373.  321,10,0,1,1,0            FILFI DSK <--> FMP.                     003820
374.  322,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    003830
375.  322,10,0,1,1,-1           FILE2 DSK <--> FMP.                     003840
376.  320,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      003850
377.  329,10,0,0,90,-1           FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK,003860
378.  329,1,0,1,90,-1            FILE9 DSK <--> SPS.                    003870
379.  320,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    003880
380.  320,0,0,0,0,0             JOB RUN COMPLETE.                       003890
381.  330,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE003900
382.  330,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M003910
383.  330,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 003920
384.  331,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK     003930
385.  331,10,0,1,1,0            FILE1 DSK <--> FMP.                     003940
386.  332,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    003950
387.  332,10,0,1,1,-1           FILE2 DSK <--> FMP.                     003960
388.  330,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10.SEC.      003970
389.  338,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   003980
390.  338,1,0,1,2,-1            FILE8 DSK <--> SPS.                     003990
391.  330,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    004000
392.  330,0,0,0,0,0             JOB RUN COMPLETE.                       004010
393.  340,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004020
394.  340,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M004030
395.  340,61,1,15,0,-300          ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T004040
396.  340,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 004050
397.  341,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    004060
398.  341,10,0,1,1,0            FILE1 DSK <--> FMP.                     004070
399.  342,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    004080
400.  342,10,0,1,1,-1           FILE2 DSK <--> FMP.                     004090
401.  340,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      004100
402.  348,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   004110
403.  348,1,0,1,2,-1            FILE8 DSK <--> SPS.                     004120
404.  340,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    004130
405.  340,0,0,0,0,0             JOB RUN COMPLETE.                       004140
406.  350,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004150
407.  350,61,1,15,0,-1          SPS COMPILATION OF SOURCE CODE-1 SEC.   004160
408.  351,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    004170
409.  351,10,0,1,1,0            FILE1 DSK <--> FMP.                     004180
410.  352,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    004190
411.  352,10,0,1,1,-1           FILE2 DSK <--> FMP.                     004200
412.  350,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      004210
413.  358,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   004220
414.  358,1,0,1,2,-1            FILE8 DSK <--> SPS.                     004230
415.  350,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    004240
416.  350,0,0,0,0,0             JOB RUN COMPLETE.                       004250
417.  360,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004260
418.  360,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M004270
419.  360,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 004280
420.  361,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    004290
421.  361,10,0,1,1,0            FILE1 DSK <--> FMP.                     004300
```

```
422.  362,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004310
423.  362,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004320
424.  360,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004330
425.  369,10,0,0,90,-1                 FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.004340
426.  369,1,0,1,90,-1                  FILE9 DSK <--> SPS.                    004350
427.  360,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.    004360
428.  360,0,0,0,0,0                   JOB RUN COMPLETE.                       004370
429.  370,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004380
430.  370,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M004390
431.  370,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEC. 004400
432.  371,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK     004410
433.  371,10,0,1,1,0                  FILE1 DSK <--> FMP.                     004420
434.  372,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004430
435.  372,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004440
436.  370,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004450
437.  378,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.   004460
438.  378,1,0,1,2,-1                  FILE8 DSK <--> SPS.                     004470
439.  370,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.    004480
440.  370,0,0,0,0,0                   JOB RUN COMPLETE.                       004490
441.  380,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004500
442.  380,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M004510
443.  380,61,1,15,0,-300               ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T004520
444.  380,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEC. 004530
445.  381,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.    004540
446.  381,10,0,1,1,0                  FILE1 DSK <--> FMP.                     004550
447.  382,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004560
448.  382,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004570
449.  380,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004580
450.  388,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.   004590
451.  388,1,0,1,2,-1                  FILE8 DSK <--> SPS.                     004600
452.  380,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.    004610
453.  380,0,0,0,0,0                   JOB RUN COMPLETE.                       004620
454.  390,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004630
455.  390,61,1,15,0,-1                SPS COMPILATION OF SOURCE CODE-1 SEC.   004640
456.  391,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.    004650
457.  391,10,0,1,1,0                  FILE1 DSK <--> FMP.                     004660
458.  392,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004670
459.  392,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004680
460.  390,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004690
461.  399,10,0,0,90,-1                 FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.004700
462.  399,1,0,1,90,-1                  FILE9 DSK <--> SPS.                    004710
463.  390,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.    004720
464.  390,0,0,0,0,0                   JOB RUN COMPLETE.                       004730
465.  400,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004740
466.  400,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M004750
467.  400,61,1,15,0,-1                SPS RECOMPILATION OF SOURCE CODE-1 SEC. 004760
468.  401,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.    004770
469.  401,10,0,1,1,0                  FILE1 DSK <--> FMP.                     004780
470.  402,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004790
471.  402,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004800
472.  400,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004810
473.  409,10,0,0,90,-1                 FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.004820
474.  409,1,0,1,90,-1                  FILE9 DSK <--> SPS.                    004830
475.  400,61,60,12,1,-1               SPS POST-PROCESSING;40 SEC-25% LOAD.    004840
476.  400,0,0,0,0,0                   JOB RUN COMPLETE.                       004850
477.  410,0,120,1,0,0                 MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004860
478.  410,61,1,15,0,-1                SPS COMPILATION OF SOURCE CODE-1 SEC.   004870
479.  411,1,0,0,1,-1                  FILE1-LOAD MODULE 15K; SPS <--> DSK.    004880
480.  411,10,0,1,1,0                  FILE1 DSK <--> FMP.                     004890
481.  412,21,0,0,1,-1                 FILE2-CONFIGATION 10K; GRF <--> DSK.    004900
482.  412,10,0,1,1,-1                 FILE2 DSK <--> FMP.                     004910
483.  410,60,10,0,2,-1                FMP FLOW CODE PROCESSING - 10 SEC.      004920
484.  418,10,0,0,2,-1                 FILE8-RESULTS FILE 60K; FMP <--> DSK.   004930
485.  418,1,0,1,2,-1                  FILE8 DSK <--> SPS.                     004940
```

```
486.  410,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.     004950
487.  410,0,0,0,0,0                  JOB RUN COMPLETE.                        004960
488.  420,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE004970
489.  420,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.    004980
490.  421,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     004990
491.  421,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005000
492.  422,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005010
493.  422,10,0,1,1,-1                FILE2 DSK <--> FMP.                      005020
494.  420,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.       005030
495.  428,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.    005040
496.  428,1,0,1,2,-1                 FILE8 DSK <--> SPS.                      005050
497.  420,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.     005060
498.  420,0,0,0,0,0                  JOB RUN COMPLETE.                        005070
499.  430,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005080
500.  430,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.    005090
501.  431,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     005100
502.  431,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005110
503.  432,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005120
504.  432,10,0,1,1,-1                FILE2 DSK <--> FMP.                      005130
505.  430,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.       005140
506.  439,10,0,0,90,-1                FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.005150
507.  439,1,0,1,90,-1                 FILE9 DSK <--> SPS.                     005160
508.  430,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.     005170
509.  430,0,0,0,0,0                  JOB HUN COMPLETE.                        005180
510.  440,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005190
511.  440,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.    005200
512.  441,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     005210
513.  441,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005220
514.  442,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005230
515.  442,10,0,1,1,-1                FILE2 DSK <--> FMP.                      005240
516.  440,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.       005250
517.  449,10,0,0,90,-1                FILE9-DEBUG DUMP 3M; FMP <--> DSK.      005260
518.  449,1,0,1,90,-1                 FILE9 DSK <--> SPS.                     005270
519.  440,61,60,12,1,-1              SPS POST-PHOCESSING;40 SEC-25% LOAD.     005280
520.  440,0,0,0,0,0                  JOB RUN COMPLETE.                        005290
521.  450,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005300
522.  450,61,1,15,0,-600               1 SEC ABORTED SPS COMPIL'N - WAIT 10 M005310
523.  450,61,1,15,0,-1               SPS RECOMPILATION OF SOURCE CODE-1 SEC. 005320
524.  451,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     005330
525.  451,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005340
526.  452,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005350
527.  452,10,0,1,1,-1                FILE2 DSK <--> FMP.                      005360
528.  450,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.       005370
529.  458,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.    005380
530.  458,1,0,1,2,-1                 FILE8 DSK <--> SPS.                      005390
531.  450,61,60,12,1,-1              SPS POST-PHOCESSING;40 SEC-25% LOAD.     005400
532.  450,0,0,0,0,0                  JOB RUN COMPLETE.                        005410
533.  460,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005420
534.  460,61,1,15,0,-600               1 SEC ABORTED SPS COMPIL'N - WAIT 10 M005430
535.  460,61,1,15,0,-300               ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T005440
536.  460,61,1,15,0,-1               SPS HECOMPILATION OF SOURCE CODE-1 SEC. 005450
537.  461,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     005460
538.  461,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005470
539.  462,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005480
540.  462,10,0,1,1,-1                FILE2 DSK <--> FMP.                      005490
541.  460,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.       005500
542.  468,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.    005510
543.  468,1,0,1,2,-1                 FILE8 DSK <--> SPS.                      005520
544.  460,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.     005530
545.  460,0,0,0,0,0                  JOB HUN COMPLETE.                        005540
546.  470,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005550
547.  470,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SFC.    005560
548.  471,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.     005570
549.  471,10,0,1,1,0                 FILE1 DSK <--> FMP.                      005580
550.  472,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.     005590
```

```
551.  472,10,0,1,1,-1           FILE2 DSK <--> FMP.                     005600
552.  470,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      005610
553.  478,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   005620
554.  478,1,0,1,2,-1            FILE8 DSK <--> SPS.                     005630
555.  470,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    005640
556.  470,0,0,0,0,0             JOB RUN COMPLETE.                       005650
557.  480,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005660
558.  480,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M005670
559.  480,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 005680
560.  481,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK     005690
561.  481,10,0,1,1,0            FILE1 DSK <--> FMP.                     005700
562.  482,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    005710
563.  482,10,0,1,1,-1           FILE2 DSK <--> FMP.                     005720
564.  480,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      005730
565.  489,10,0,0,90,-1           FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.005740
566.  489,1,0,1,90,-1            FILE9 DSK <--> SPS.                    005750
567.  480,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    005760
568.  480,0,0,0,0,0             JOB RUN COMPLETE.                       005770
569.  490,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005780
570.  490,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M005790
571.  490,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 005800
572.  491,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    005810
573.  491,10,0,1,1,0            FILE1 DSK <--> FMP.                     005820
574.  492,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    005830
575.  492,10,0,1,1,-1           FILE2 DSK <--> FMP.                     005840
576.  490,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      005850
577.  498,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   005860
578.  498,1,0,1,2,-1            FILE8 DSK <--> SPS.                     005870
579.  490,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    005880
580.  490,0,0,0,0,0             JOB RUN COMPLETE.                       005890
581.  500,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE005900
582.  500,61,1,15,0,-600          1 SEC ABORTED SPS COMPIL'N - WAIT 10 M005910
583.  500,61,1,15,0,-300          ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T005920
584.  500,61,1,15,0,-1          SPS RECOMPILATION OF SOURCE CODE-1 SEC. 005930
585.  501,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    005940
586.  501,10,0,1,1,0            FILE1 DSK <--> FMP.                     005950
587.  502,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    005960
588.  502,10,0,1,1,-1           FILE2 DSK <--> FMP.                     005970
589.  500,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      005980
590.  508,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   005990
591.  508,1,0,1,2,-1            FILE8 DSK <--> SPS.                     006000
592.  500,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    006010
593.  500,0,0,0,0,0             JOB RUN COMPLETE.                       006020
594.  510,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006030
595.  510,61,1,15,0,-1          SPS COMPILATION OF SOURCE CODE-1 SEC.   006040
596.  511,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    006050
597.  511,10,0,1,1,0            FILE1 DSK <--> FMP.                     006060
598.  512,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    006070
599.  512,10,0,1,1,-1           FILE2 DSK <--> FMP.                     006080
600.  510,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      006090
601.  518,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   006100
602.  518,1,0,1,2,-1            FILE8 DSK <--> SPS.                     006110
603.  510,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    006120
604.  510,0,0,0,0,0             JOB RUN COMPLETE.                       006130
605.  520,0,120,1,0,0           MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006140
606.  520,61,1,15,0,-1          SPS COMPILATION OF SOURCE CODE-1 SEC.   006150
607.  521,1,0,0,1,-1            FILE1-LOAD MODULE 15K; SPS <--> DSK.    006160
608.  521,10,0,1,1,0            FILE1 DSK <--> FMP.                     006170
609.  522,21,0,0,1,-1           FILE2-CONFIGATION 10K; GRF <--> DSK.    006180
610.  522,10,0,1,1,-1           FILE2 DSK <--> FMP.                     006190
611.  520,60,10,0,2,-1          FMP FLOW CODE PROCESSING - 10 SEC.      006200
612.  528,10,0,0,2,-1           FILE8-RESULTS FILE 60K; FMP <--> DSK.   006210
613.  528,1,0,1,2,-1            FILE8 DSK <--> SPS.                     006220
614.  520,61,60,12,1,-1         SPS POST-PROCESSING;40 SEC-25% LOAD.    006230
615.  520,0,0,0,0,0             JOB RUN COMPLETE.                       006240
```

```
616.  530,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006250
617.  530,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.   006260
618.  531,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.    006270
619.  531,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006280
620.  532,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006290
621.  532,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006300
622.  530,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.      006310
623.  539,10,0,0,90,-1                FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.006320
624.  539,1,0,1,90,-1                 FILE9 DSK <--> SPS.                    006330
625.  530,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    006340
626.  530,0,0,0,0,0                  JOB RUN COMPLETE.                       006350
627.  540,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006360
628.  540,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.   006370
629.  541,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.    006380
630.  541,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006390
631.  542,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006400
632.  542,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006410
633.  540,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.      006420
634.  549,10,0,0,90,-1                FILE9-DEBUG DUMP 3M; FMP <--> DSK.     006430
635.  549,1,0,1,90,-1                 FILE9 DSK <--> SPS.                    006440
636.  540,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    006450
637.  540,0,0,0,0,0                  JOB RUN COMPLETE.                       006460
638.  550,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006470
639.  550,61,1,15,0,-600                1 SEC ABORTED SPS COMPIL'N - WAIT 10 M006480
640.  550,61,1,15,0,-1               SPS RECOMPILATION OF SOURCE CODE-1 SEC. 006490
641.  551,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.    006500
642.  551,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006510
643.  552,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006520
644.  552,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006530
645.  550,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.      006540
646.  558,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.   006550
647.  558,1,0,1,2,-1                 FILE8 DSK <--> SPS.                     006560
648.  550,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    006570
649.  550,0,0,0,0,0                  JOB RUN COMPLETE.                       006580
650.  560,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006590
651.  560,61,1,15,0,-600               1 SEC ABORTED SPS COMPIL'N - WAIT 10 M006600
652.  560,61,1,15,0,-300               ABORTED COMPIL'N AGAIN - WAIT 5 MIN, T006610
653.  560,61,1,15,0,-1               SPS RECOMPILATION OF SOURCE CODE-1 SEC. 006620
654.  561,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.    006630
655.  561,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006640
656.  562,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006650
657.  562,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006660
658.  560,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.      006670
659.  568,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.   006680
660.  568,1,0,1,2,-1                 FILE8 DSK <--> SPS.                     006690
661.  560,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    006700
662.  560,0,0,0,0,0                  JOB RUN COMPLETE.                       006710
663.  570,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006720
664.  570,61,1,15,0,-1               SPS COMPILATION OF SOURCE CODE-1 SEC.   006730
665.  571,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK.    006740
666.  571,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006750
667.  572,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006760
668.  572,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006770
669.  570,60,10,0,2,-1               FMP FLOW CODE PROCESSING - 10 SEC.      006780
670.  578,10,0,0,2,-1                FILE8-RESULTS FILE 60K; FMP <--> DSK.   006790
671.  578,1,0,1,2,-1                 FILE8 DSK <--> SPS.                     006800
672.  570,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    006810
673.  570,0,0,0,0,0                  JOB RUN COMPLETE.                       006820
674.  580,0,120,1,0,0                MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006830
675.  580,61,1,15,0,-600               1 SEC ABORTED SPS COMPIL'N - WAIT 10 M006840
676.  580,61,1,15,0,-1               SPS RECOMPILATION OF SOURCE CODE-1 SEC. 006850
677.  581,1,0,0,1,-1                 FILE1-LOAD MODULE 15K; SPS <--> DSK     006860
678.  581,10,0,1,1,0                 FILE1 DSK <--> FMP.                     006870
679.  582,21,0,0,1,-1                FILE2-CONFIGATION 10K; GRF <--> DSK.    006880
680.  582,10,0,1,1,-1                FILE2 DSK <--> FMP.                     006890
```

```
681.  580,60,10,0,2,-1              FMP FLOW CODE PROCESSING - 10 SEC.       006900
682.  589,10,0,0,90,-1               FILE9-DEBUG DUMP FILE 3M; FMP <--> DSK.006910
683.  589,1,0,1,90,-1                FILE9 DSK <--> SPS.                     006920
684.  580,61,60,12,1,-1             SPS POST-PROCESSING;40 SEC-25% LOAD.     006930
685.  580,0,0,0,0,0                 JOB RUN COMPLETE.                        006940
686.  590,0,120,1,0,0               MODEL 1-INTERARRIVAL TIME MEAN OF 120 SE006950
687.  590,61,1,15,0,-600              1 SEC ABORTED SPS COMPIL'N - WAIT 10 M006960
688.  590,61,1,15,0,-1              SPS RECOMPILATION OF SOURCE CODE-1 SEC. 006970
689.  591,1,0,0,1,-1                FILE1-LOAD MODULE 15K; SPS <--> DSK.     006980
690.  591,10,0,1,1,0                FILE1 DSK <--> FMP.                      006990
691.  592,21,0,0,1,-1               FILE2-CONFIGATION 10K; GRF <--> DSK.     007000
692.  592,10,0,1,1,-1               FILE2 DSK <--> FMP.                      007010
693.  590,60,10,0,2,-1              FMP FLOW CODE PROCESSING - 10 SEC.       007020
694.  598,10,0,0,2,-1               FILE8-RESULTS FILE 60K; FMP <--> DSK.    007030
695.  598,1,0,1,2,-1                FILE8 DSK <--> SPS.                      007040
696.  590,61,60,12,1,-1              SPS POST-PROCESSING;40 SEC-25% LOAD.    007050
697.  590,0,0,0,0,0                  JOB RUN COMPLETE.                       007060
698.  610,0,300,2,0,0              MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007070
699.  610,61,1,25,0,-600           1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 007080
700.  610,61,1,25,0,-1             SPS RECOMPILATION OF SOURCE CODE - 1 SEC.007090
701.  611,1,0,0,2,-1               FILE1-LOAD MODULE 60K; SPS --> DSK.      007100
702.  611,10,0,1,2,0               FILE1 DSK --> FMP.                       007110
703.  612,21,0,0,2,-1              FILE2-CONFIGURATION 50K; GRF --> DSK.    007120
704.  612,10,0,1,2,0               FILE2 DSK --> FMP.                       007130
705.  613,1,0,0,19,-1              FILE3-GRID 600K; SPS --> DSK.            007140
706.  613,10,0,1,19,-1             FILE3 DSK --> FMP.                       007150
707.  610,60,60,0,3,-1             FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007160
708.  619,10,0,0,125,-1            FILE9-DEBUG DUMP 4M; FMP --> DSK.        007170
709.  619,1,0,1,125,0              FILE9  DSK --> SPS.                      007180
710.  619,10,0,0,125,-1            FILE9 ANOTHER 4M WORDS; FMP --> DSK.     007190
711.  619,1,0,1,125,-1             FILE9 DSK --> SPS.                       007200
712.  610,0,0,0,0,0                JOB RUN COMPLETED.                       007210
713.  620,0,300,1,0,0              MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007220
714.  620,61,1,25,0,-1             SPS COMPILATION OF SOURCE CODE - 1 SEC   007230
715.  621,1,0,0,2,-1               FILE1-LOAD MODULE 60K; SPS --> DSK.      007240
716.  621,10,0,1,2,0               FILE1 DSK --> FMP.                       007250
717.  622,20,0,1,2,-1              FILE2-CONFIGURATION 50K; GRF --> SPS.    007260
718.  620,61,80,20,1,-1            SPS CONFIGUR'N MANIPULA'N;70 SEC-30% LOAD007270
719.  622,1,0,0,2,-1               FILE2-CONFIGURATION 50K; SPS --> DSK.    007280
720.  622,10,0,1,2,0               FILE2 DSK --> FMP.                       007290
721.  623,1,0,0,19,-1              FILE3-GRID 600K; SPS --> DSK.            007300
722.  623,10,0,1,19,-1             FILE3 DSK --> FMP.                       007310
723.  620,60,60,0,3,-1             FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007320
724.  628,10,0,0,6,-1              FILE8-RESULTS FILE 180K; FMP --> DSK.    007330
725.  628,1,0,1,6,-1               FILE8  DSK --> SPS.                      007340
726.  620,61,180,35,1,-1           SPS POST PROCESSING-120 SEC;70% LOAD.    007350
727.  627,20,0,0,4,-1              FILE7-OUTPUT FILE 120K; SPS --> GRF.     007360
728.  620,0,0,0,0,0                JOB RUN COMPLETED.                       007370
729.  630,0,300,1,0,0              MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007380
730.  630,61,1,25,0,-600           1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 007390
731.  630,61,1,25,0,-300           ABORTED COMPILATION AGAIN-WAIT 5 MIN.    007400
732.  630,61,1,25,0,-1             SPS RECOMPILATION OF SOURCE CODE - 1 SEC.007410
733.  631,1,0,0,2,-1               FILE1-LOAD MODULE 60K; SPS --> DSK.      007420
734.  631,10,0,1,2,0               FILE1 DSK --> FMP.                       007430
735.  632,21,0,0,2,-1              FILE2-CONFIGURATION 50K; GRF --> DSK.    007440
736.  632,10,0,1,2,0               FILE2 DSK --> FMP.                       007450
737.  633,1,0,0,19,-1              FILE3-GRID 600K; SPS --> DSK.            007460
738.  633,10,0,1,19,-1             FILE3 DSK --> FMP.                       007470
739.  630,60,60,0,3,-1             FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007480
740.  638,10,0,0,12,-1             FILE8-RESULTS FILE 360K; FMP --> DSK.    007490
741.  638,1,0,1,12,-1              FILE8  DSK --> SPS.                      007500
742.  630,61,360,35,1,-1           SPS POST PROCESSING-240 SEC;70% LOAD.    007510
743.  637,20,0,0,8,-1              FILE7-OUTPUT FILE 240K; SPS --> GRF.     007520
744.  630,0,0,0,0,0                JOB RUN COMPLETED.                       007530
745.  640,0,300,1,0,0              MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007540
```

```
746.  640,61,1,25,0,-600          1 SEC ABORTED COMP'LN-WAIT 10 MIN.         007550
747.  640,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.007560
748.  641,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.        007570
749.  641,10,0,1,2,0              FILE1 DSK --> FMP.                         007580
750.  642,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.      007590
751.  642,10,0,1,2,0              FILE2 DSK --> FMP.                         007600
752.  643,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.              007610
753.  643,10,0,1,19,-1            FILE3 DSK --> FMP.                         007620
754.  640,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007630
755.  648,10,0,0,6,-1             FILE8-RESULTS FILE 180K; FMP --> DSK.      007640
756.  648,1,0,1,6,-1              FILE8  DSK --> SPS.                        007650
757.  640,61,180,35,1,-1          SPS POST PROCESSING-120 SEC;70% LOAD.      007660
758.  647,20,0,0,4,-1             FILE7-OUTPUT FILE 120K; SPS --> GRF.       007670
759.  640,0,0,0,0,0               JOB RUN COMPLETED.                         007680
760.  650,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007690
761.  650,61,1,25,0,-600          1 SEC ABORTED COMPL'N - WAIT 10 MIN.       007700
762.  650,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.007710
763.  651,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.        007720
764.  651,10,0,1,2,0              FILE1 DSK --> FMP.                         007730
765.  652,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.      007740
766.  652,10,0,1,2,0              FILE2 DSK --> FMP.                         007750
767.  653,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.              007760
768.  653,10,0,1,19,-1            FILE3 DSK --> FMP.                         007770
769.  650,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007780
770.  658,10,0,0,6,-1             FILE8-RESULTS FILE 180K; FMP --> DSK.      007790
771.  658,1,0,1,6,-1              FILE8  DSK --> SPS.                        007800
772.  650,61,180,35,1,-1          SPS POST PROCESSING-120 SEC;70% LOAD.      007810
773.  657,20,0,0,4,-1             FILE7-OUTPUT FILE 120K; SPS --> GRF.       007820
774.  650,0,0,0,0,0               JOB RUN COMPLETED.                         007830
775.  660,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007840
776.  660,61,1,25,0,-600          1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 007850
777.  660,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.007860
778.  661,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.        007870
779.  661,10,0,1,2,0              FILE1 DSK --> FMP.                         007880
780.  662,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.      007890
781.  662,10,0,1,2,0              FILE2 DSK --> FMP.                         007900
782.  663,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.              007910
783.  663,10,0,1,19,-1            FILE3 DSK --> FMP.                         007920
784.  660,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI007930
785.  669,10,0,0,125,-1           FILE9-DEBUG DUMP 4M; FMP --> DSK.          007940
786.  669,1,0,1,125,0             FILE9  DSK --> SPS.                        007950
787.  669,10,0,0,125,-1           FILE9 ANOTHER 4M WORDS; FMP --> DSK.       007960
788.  669,1,0,1,125,-1            FILE9 DSK --> SPS.                         007970
789.  660,0,0,0,0,0               JOB RUN COMPLETED.                         007980
790.  670,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC007990
791.  670,61,1,25,0,-1            SPS COMPILATION OF SOURCE CODE - 1 SEC     008000
792.  671,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.        008010
793.  671,10,0,1,2,0              FILE1 DSK --> FMP.                         008020
794.  672,20,0,1,2,-1             FILE2-CONFIGURATION 50K; GRF --> SPS.      008030
795.  670,61,80,20,1,-1           SPS CONFIGUR'N MANIPULA'N;70 SEC-30% LOAD008040
796.  672,1,0,0,2,-1              FILE2-CONFIGURATION 50K; SPS --> DSK.      008050
797.  672,10,0,1,2,0              FILE2 DSK --> FMP.                         008060
798.  673,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.              008070
799.  673,10,0,1,19,-1            FILE3 DSK --> FMP.                         008080
800.  670,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008090
801.  679,10,0,0,125,-1           FILE9-DEBUG DUMP 4M; FMP --> DSK.          008100
802.  679,1,0,1,125,0             FILE9  DSK --> SPS.                        008110
803.  679,10,0,0,125,-1           FILE9 - ANOTHER 4N WORDS FMP --> DSK.      008120
804.  679,1,0,1,125,-1            FILE9 DSK --> SPS.                         008130
805.  670,0,0,0,0,0               JOB RUN COMPLETED.                         008140
806.  680,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008150
807.  680,61,1,25,0,-600          1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 008160
808.  680,61,1,25,0,-300          ABORTED COMPILATION AGAIN-WAIT 5 MIN.      008170
809.  680,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.008180
810.  681,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.        008190
```

```
811.  681,10,0,1,2,0              FILE1 DSK --> FMP.                       008200
812.  682,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.    008210
813.  682,10,0,1,2,0              FILE2 DSK --> FMP.                       008220
814.  683,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.            008230
815.  683,10,0,1,19,-1            FILE3 DSK --> FMP.                       008240
816.  680,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008250
817.  688,10,0,0,6,-1             FILE8-RESULTS FILE 180K; FMP --> DSK.    008260
818.  688,1,0,1,6,-1              FILE8  DSK --> SPS.                      008270
819.  680,61,180,35,1,-1          SPS POST PROCESSING-120 SEC;70% LOAD.    008280
820.  687,20,0,0,4,-1             FILE7-OUTPUT FILE 120K; SPS --> GRF.     008290
821.  680,0,0,0,0,0               JOB RUN COMPLETED.                       008300
822.  690,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008310
823.  690,61,1,25,0,-600          1 SEC ABORTED COMP'LN-WAIT 10 MIN.       008320
824.  690,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.008330
825.  691,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.      008340
826.  691,10,0,1,2,0              FILE1 DSK --> FMP.                       008350
827.  692,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.    008360
828.  692,10,0,1,2,0              FILE2 DSK --> FMP.                       008370
829.  693,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.            008380
830.  693,10,0,1,19,-1            FILE3 DSK --> FMP.                       008390
831.  690,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008400
832.  698,10,0,0,6,-1             FILE8-RESULTS FILE 180K; FMP --> DSK.    008410
833.  698,1,0,1,6,-1              FILE8  DSK --> SPS.                      008420
834.  690,61,180,35,1,-1          SPS POST PROCESSING-120 SEC;70% LOAD.    008430
835.  697,20,0,0,4,-1             FILE7-OUTPUT FILE 120K; SPS --> GRF.     008440
836.  690,0,0,0,0,0               JOB RUN COMPLETED.                       008450
837.  700,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008460
838.  700,61,1,25,0,-600          1 SEC ABORTED COMPL'N - WAIT 10 MIN.     008470
839.  700,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.008480
840.  701,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.      008490
841.  701,10,0,1,2,0              FILE1 DSK --> FMP.                       008500
842.  702,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.    008510
843.  702,10,0,1,2,0              FILE2 DSK --> FMP.                       008520
844.  703,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.            008530
845.  703,10,0,1,19,-1            FILE3 DSK --> FMP.                       008540
846.  700,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008550
847.  708,10,0,0,12,-1            FILE8-RESULTS FILE 360K; FMP --> DSK.    008560
848.  708,1,0,1,12,-1             FILE8  DSK --> SPS.                      008570
849.  700,61,360,35,1,-1          SPS POST PROCESSING-240 SEC;70% LOAD.    008580
850.  707,20,0,0,8,-1             FILE7-OUTPUT FILE 240K; SPS --> GRF.     008590
851.  700,0,0,0,0,0               JOB RUN COMPLETED.                       008600
852.  710,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008610
853.  710,61,1,25,0,-600          1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 008620
854.  710,61,1,25,0,-1            SPS RECOMPILATION OF SOURCE CODE - 1 SEC.008630
855.  711,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.      008640
856.  711,10,0,1,2,0              FILE1 DSK --> FMP.                       008650
857.  712,21,0,0,2,-1             FILE2-CONFIGURATION 50K; GRF --> DSK.    008660
858.  712,10,0,1,2,0              FILE2 DSK --> FMP.                       008670
859.  713,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.            008680
860.  713,10,0,1,19,-1            FILE3 DSK --> FMP.                       008690
861.  710,60,60,0,3,-1            FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008700
862.  719,10,0,0,125,-1           FILE9-DEBUG DUMP 4M; FMP --> DSK.        008710
863.  719,1,0,1,125,0             FILE9  DSK --> SPS.                      008720
864.  719,10,0,0,125,-1           FILE9 ANOTHER 4M WORDS; FMP --> DSK.     008730
865.  719,1,0,1,125,-1            FILE9 DSK --> SPS.                       008740
866.  710,0,0,0,0,0               JOB RUN COMPLETED.                       008750
867.  720,0,300,1,0,0             MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008760
868.  720,61,1,25,0,-1            SPS COMPILATION OF SOURCE CODE - 1 SEC.  008770
869.  721,1,0,0,2,-1              FILE1-LOAD MODULE 60K; SPS --> DSK.      008780
870.  721,10,0,1,2,0              FILE1 DSK --> FMP.                       008790
871.  722,20,0,1,2,-1             FILE2-CONFIGURATION 50K; GRF --> SPS.    008800
872.  720,61,80,20,1,-1           SPS CONFIGUR'N MANIPULA'N;70 SEC-30% LOAD008810
873.  722,1,0,0,2,-1              FILE2-CONFIGURATION 50K; SPS --> DSK.    008820
874.  722,10,0,1,2,0              FILE2 DSK --> FMP.                       008830
875.  723,1,0,0,19,-1             FILE3-GRID 600K; SPS --> DSK.            008840
```

```
876.  723,10,0,1,19,-1              FILE3 DSK --> FMP.                           008850
877.  720,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI008860
878.  728,10,0,0,6,-1               FILE8-RESULTS FILE 180K; FMP --> DSK.        008870
879.  728,1,0,1,6,-1                FILE8  DSK --> SPS.                          008880
880.  720,61,180,35,1,-1            SPS POST PROCESSING-120 SEC;70% LOAD.        008890
881.  727,20,0,0,4,-1               FILE7-OUTPUT FILE 120K; SPS --> GRF.         008900
882.  720,0,0,0,0,0                 JOB RUN COMPLETED.                           008910
883.  730,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC008920
884.  730,61,1,25,0,-600            1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 008930
885.  730,61,1,25,0,-300            ABORTED COMPILATION AGAIN-WAIT 5 MIN.        008940
886.  730,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.008950
887.  731,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.          008960
888.  731,10,0,1,2,0                FILE1 DSK --> FMP.                           008970
889.  732,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.        008980
890.  732,10,0,1,2,0                FILE2 DSK --> FMP.                           008990
891.  733,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.                009000
892.  733,10,0,1,19,-1              FILE3 DSK --> FMP.                           009010
893.  730,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009020
894.  738,10,0,0,6,-1               FILE8-RESULTS FILE 180K; FMP --> DSK.        009030
895.  738,1,0,1,6,-1                FILE8  DSK --> SPS.                          009040
896.  730,61,180,35,1,-1            SPS POST PROCESSING-120 SEC;70% LOAD.        009050
897.  737,20,0,0,4,-1               FILE7-OUTPUT FILE 120K; SPS --> GRF.         009060
898.  730,0,0,0,0,0                 JOB RUN COMPLETED.                           009070
899.  740,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009080
900.  740,61,1,25,0,-600            1 SEC ABORTED COMP'LN-WAIT 10 MIN.           009090
901.  740,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.009100
902.  741,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.          009110
903.  741,10,0,1,2,0                FILE1 DSK --> FMP.                           009120
904.  742,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.        009130
905.  742,10,0,1,2,0                FILE2 DSK --> FMP.                           009140
906.  743,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.                009150
907.  743,10,0,1,19,-1              FILE3 DSK --> FMP.                           009160
908.  740,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009170
909.  748,10,0,0,12,-1              FILE8-RESULTS FILE 360K; FMP --> DSK.        009180
910.  748,1,0,1,12,-1               FILE8  DSK --> SPS.                          009190
911.  740,61,360,35,1,-1            SPS POST PROCESSING-240 SEC;70% LOAD.        009200
912.  747,20,0,0,8,-1               FILE7-OUTPUT FILE 280K; SPS --> GRF.         009210
913.  740,0,0,0,0,0                 JOB RUN COMPLETED.                           009220
914.  750,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009230
915.  750,61,1,25,0,-600            1 SEC ABORTED COMPL'N - WAIT 10 MIN.         009240
916.  750,61,1,25,0,-300            ABORTED COMPIL'N AGAIN - WAIT 5 MIN...       009250
917.  750,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.009260
918.  751,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.          009270
919.  751,10,0,1,2,0                FILE1 DSK --> FMP.                           009280
920.  752,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.        009290
921.  752,10,0,1,2,0                FILE2 DSK --> FMP.                           009300
922.  753,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.                009310
923.  753,10,0,1,19,-1              FILE3 DSK --> FMP.                           009320
924.  750,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009330
925.  758,10,0,0,6,-1               FILE8-RESULTS FILE 180K; FMP --> DSK.        009340
926.  758,1,0,1,6,-1                FILE8  DSK --> SPS.                          009350
927.  750,61,180,35,1,-1            SPS POST PROCESSING-120 SEC;70% LOAD.        009360
928.  757,20,0,0,4,-1               FILE7-OUTPUT FILE 120K; SPS --> GRF.         009370
929.  750,0,0,0,0,0                 JOB RUN COMPLETED.                           009380
930.  760,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009390
931.  760,61,1,25,0,-600            1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 009400
932.  760,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.009410
933.  761,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.          009420
934.  761,10,0,1,2,0                FILE1 DSK --> FMP.                           009430
935.  762,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.        009440
936.  762,10,0,1,2,0                FILE2 DSK --> FMP.                           009450
937.  763,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.                009460
938.  763,10,0,1,19,-1              FILE3 DSK --> FMP.                           009470
939.  760,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009480
940.  769,10,0,0,125,-1             FILE9-DEBUG DUMP 4M; FMP --> DSK.            009490
```

```
 941.  769,1,0,1,125,0                         FILE9  DSK --> SPS.                      009500
 942.  769,10,0,0,125,-1                       FILE9 ANOTHER 4M WORDS; FMP --> DSK.     009510
 943.  769,1,0,1,125,-1                        FILE9 DSK --> SPS.                       009520
 944.  760,0,0,0,0,0                           JOB RUN COMPLETED.                       009530
 945.  770,0,300,1,0,0                         MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009540
 946.  770,61,1,25,0,-1                        SPS COMPILATION OF SOURCE CODE - 1 SEC   009550
 947.  771,1,0,0,2,-1                          FILE1-LOAD MODULE 60K; SPS --> DSK.      009560
 948.  771,10,0,1,2,0                          FILE1 DSK --> FMP.                       009570
 949.  772,20,0,1,2,-1                         FILE2-CONFIGURATION 50K; GRF --> SPS.    009580
 950.  770,61,80,20,1,-1                       SPS CONFIGUR'N MANIPULA'N;70 SEC-30% LOAD009590
 951.  772,1,0,0,2,-1                          FILE2-CONFIGURATION 50K; SPS --> DSK.    009600
 952.  772,10,0,1,2,0                          FILE2 DSK --> FMP.                       009610
 953.  773,1,0,0,19,-1                         FILE3-GRID 600K; SPS --> DSK.            009620
 954.  773,10,0,1,19,-1                        FILE3 DSK --> FMP.                       009630
 955.  770,60,60,0,3,-1                        FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009640
 956.  778,10,0,0,6,-1                         FILE8-RESULTS FILE 180K; FMP --> DSK.    009650
 957.  778,1,0,1,6,-1                          FILE8  DSK --> SPS.                      009660
 958.  770,61,180,35,1,-1                      SPS POST PROCESSING-120 SEC;70% LOAD.    009670
 959.  777,20,0,0,4,-1                         FILE7-OUTPUT FILE 120K; SPS --> GRF.     009680
 960.  770,0,0,0,0,0                           JOB RUN COMPLETED.                       009690
 961.  780,0,300,1,0,0                         MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009700
 962.  780,61,1,25,0,-600                      1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 009710
 963.  780,61,1,25,0,-300                      ABORTED COMPILATION AGAIN-WAIT 5 MIN.    009720
 964.  780,61,1,25,0,-1                        SPS RECOMPILATION OF SOURCE CODE - 1 SFC.009730
 965.  781,1,0,0,2,-1                          FILE1-LOAD MODULE 60K; SPS --> DSK.      009740
 966.  781,10,0,1,2,0                          FILE1 DSK --> FMP.                       009750
 967.  782,21,0,0,2,-1                         FILE2-CONFIGURATION 50K; GRF --> DSK.    009760
 968.  782,10,0,1,2,0                          FILE2 DSK --> FMP.                       009770
 969.  783,1,0,0,19,-1                         FILE3-GRID 600K; SPS --> DSK.            009780
 970.  783,10,0,1,19,-1                        FILE3 DSK --> FMP.                       009790
 971.  780,60,60,0,3,-1                        FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009800
 972.  788,10,0,0,6,-1                         FILE8-RESULTS FILE 180K; FMP --> DSK.    009810
 973.  788,1,0,1,6,-1                          FILE8  DSK --> SPS.                      009820
 974.  780,61,180,35,1,-1                      SPS POST PROCESSING-120 SEC;70% LOAD.    009830
 975.  787,20,0,0,4,-1                         FILE7-OUTPUT FILE 120K; SPS --> GRF.     009840
 976.  780,0,0,0,0,0                           JOB RUN COMPLETED.                       009850
 977.  790,0,300,1,0,0                         MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC009860
 978.  790,61,1,25,0,-600                      1 SEC ABORTED COMP'LN-WAIT 10 MIN.       009870
 979.  790,61,1,25,0,-1                        SPS RECOMPILATION OF SOURCE CODE - 1 SEC.009880
 980.  791,1,0,0,2,-1                          FILE1-LOAD MODULE 60K; SPS --> DSK.      009890
 981.  791,10,0,1,2,0                          FILE1 DSK --> FMP.                       009900
 982.  792,21,0,0,2,-1                         FILE2-CONFIGURATION 50K; GRF --> DSK.    009910
 983.  792,10,0,1,2,0                          FILE2 DSK --> FMP.                       009920
 984.  793,1,0,0,19,-1                         FILE3-GRID 600K; SPS --> DSK.            009930
 985.  793,10,0,1,19,-1                        FILE3 DSK --> FMP.                       009940
 986.  790,60,60,0,3,-1                        FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI009950
 987.  798,10,0,0,6,-1                         FILE8-RESULTS FILE 180K; FMP --> DSK.    009960
 988.  798,1,0,1,6,-1                          FILE8  DSK --> SPS.                      009970
 989.  790,61,180,35,1,-1                      SPS POST PROCESSING-120 SEC;70% LOAD.    009980
 990.  797,20,0,0,4,-1                         FILE7-OUTPUT FILE 120K; SPS --> GRF.     009990
 991.  790,0,0,0,0,0                           JOB RUN COMPLETED.                       010000
 992.  800,0,300,1,0,0                         MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC010010
 993.  800,61,1,25,0,-600                      1 SEC ABORTED COMPL'N - WAIT 10 MIN.     010020
 994.  800,61,1,25,0,-1                        SPS RECOMPILATION OF SOURCE CODF - 1 SEC.010030
 995.  801,1,0,0,2,-1                          FILE1-LOAD MODULE 60K; SPS --> DSK.      010040
 996.  801,10,0,1,2,0                          FILE1 DSK --> FMP.                       010050
 997.  802,21,0,0,2,-1                         FILE2-CONFIGURATION 50K; GRF --> DSK.    010060
 998.  802,10,0,1,2,0                          FILE2 DSK --> FMP.                       010070
 999.  803,1,0,0,19,-1                         FILE3-GRID 600K; SPS --> DSK.            010080
1000.  803,10,0,1,19,-1                        FILE3 DSK --> FMP.                       010090
1001.  800,60,60,0,3,-1                        FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI010100
1002.  808,10,0,0,6,-1                         FILE8-RESULTS FILE 180K; FMP --> DSK.    010110
1003.  808,1,0,1,6,-1                          FILE8  DSK --> SPS.                      010120
1004.  800,61,180,35,1,-1                      SPS POST PROCESSING-120 SEC;70% LOAD.    010130
1005.  807,20,0,0,4,-1                         FILE7-OUTPUT FILE 120K; SPS --> GRF.     010140
1006.  800,0,0,0,0,0                           JOB RUN COMPLETED.                       010150
```

```
1007.  810,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC010160
1008.  810,61,1,25,0,-600            1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 010170
1009.  810,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.010180
1010.  811,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.      010190
1011.  811,10,0,1,2,0                FILE1 DSK --> FMP.                       010200
1012.  812,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.    010210
1013.  812,10,0,1,2,0                FILE2 DSK --> FMP.                       010220
1014.  813,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.            010230
1015.  813,10,0,1,19,-1              FILE3 DSK --> FMP.                       010240
1016.  810,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI010250
1017.  819,10,0,0,125,-1             FILE9-DEBUG DUMP 4M; FMP --> DSK.        010260
1018.  819,1,0,1,125,0               FILE9  DSK --> SPS.                      010270
1019.  819,10,0,0,125,-1             FILE9 ANOTHER 4M WORDS; FMP --> DSK.     010280
1020.  819,1,0,1,125,-1              FILE9 DSK --> SPS.                       010290
1021.  810,0,0,0,0,0                 JOB RUN COMPLETED.                       010300
1022.  820,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC010310
1023.  820,61,1,25,0,-1              SPS COMPILATION OF SOURCE CODE - 1 SEC   010320
1024.  821,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.      010330
1025.  821,10,0,1,2,0                FILE1 DSK --> FMP.                       010340
1026.  822,20,0,1,2,-1               FILE2-CONFIGURATION 50K; GRF --> SPS.    010350
1027.  820,61,80,20,1,-1             SPS CONFIGUR'N MANIPULA'N;70 SEC-30% LOAD010360
1028.  822,1,0,0,2,-1                FILE2-CONFIGURATION 50K; SPS --> DSK.    010370
1029.  822,10,0,1,2,0                FILE2 DSK --> FMP.                       010380
1030.  823,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.            010390
1031.  823,10,0,1,19,-1              FILE3 DSK --> FMP.                       010400
1032.  820,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI010410
1033.  828,10,0,0,6,-1               FILE8-RESULTS FILE 180K; FMP --> DSK.    010420
1034.  828,1,0,1,6,-1                FILE8  DSK --> SPS.                      010430
1035.  820,61,180,35,1,-1            SPS POST PROCESSING-120 SEC;70% LOAD.    010440
1036.  827,20,0,0,4,-1               FILE7-OUTPUT FILE 120K; SPS --> GRF.     010450
1037.  820,0,0,0,0,0                 JOB RUN COMPLETED.                       010460
1038.  830,0,300,1,0,0               MODEL 2-INTERARRIVAL TIME MEAN OF 300 SEC010470
1039.  830,61,1,25,0,-600            1 SEC ABORTED SPS COMP'LN - WAIT 10 MIN. 010480
1040.  830,61,1,25,0,-300            ABORTED COMPILATION AGAIN-WAIT 5 MIN.    010490
1041.  830,61,1,25,0,-1              SPS RECOMPILATION OF SOURCE CODE - 1 SEC.010500
1042.  831,1,0,0,2,-1                FILE1-LOAD MODULE 60K; SPS --> DSK.      010510
1043.  831,10,0,1,2,0                FILE1 DSK --> FMP.                       010520
1044.  832,21,0,0,2,-1               FILE2-CONFIGURATION 50K; GRF --> DSK.    010530
1045.  832,10,0,1,2,0                FILE2 DSK --> FMP.                       010540
1046.  833,1,0,0,19,-1               FILE3-GRID 600K; SPS --> DSK.            010550
1047.  833,10,0,1,19,-1              FILE3 DSK --> FMP.                       010560
1048.  830,60,60,0,3,-1              FMP FLOW CODE PROCESSING-60SEC;AWAIT 3 FI010570
1049.  838,10,0,0,12,-1              FILE8-RESULTS FILE 360K; FMP --> DSK.    010580
1050.  838,1,0,1,12,-1               FILE8  DSK --> SPS.                      010590
1051.  830,61,360,35,1,-1            SPS POST PROCESSING-240 SEC;70% LOAD.    010600
1052.  837,20,0,0,8,-1               FILE7-OUTPUT FILE 240K; SPS --> GRF.     010610
1053.  830,0,0,0,0,0                 JOB RUN COMPLETED.                       010620
1054.  850,0,1200,2,0,0              MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.010630
1055.  851,1,0,0,4,-1                FILE1-LOAD MODULE 120K; SPS --> DSK.     010640
1056.  851,10,0,1,4,0                FILE1 DSK --> FMP.                       010650
1057.  852,20,0,1,4,-1               FILE2-PATCH AND GRID SETUP 100K;GRF-->SPS010660
1058.  850,61,360,35,2,-1            SPS PATCH AND GRID PROCESSING-240S.70%   010670
1059.  853,1,0,0,46,-1               FILE3-RESULTING TRANSFORMED FILE SPS-->DS010680
1060.  853,10,0,1,46,0               FILE3 DSK --> FMP.                       010690
1061.  853,1,0,0,46,-1               REST OF FILE3- 3M TOTAL; SPS-->DSK.      010700
1062.  853,10,0,1,46,-1              REST OF FILE3 DSK --> FMP.               010710
1063.  850,60,60,0,2,-1              FMP FLOW CODE PROCESSING - 60 SEC.       010720
1064.  858,10,0,0,3,-1               FILE8-RESULTS FILE 90K; FMP --> DSK.     010730
1065.  858,1,0,1,3,-1                FILE8 DSK --> SPS.                       010740
1066.  850,61,300,30,1,-1            SPS POST-PROCESSING;200 SEC.60%LOAD.     010750
1067.  857,20,0,0,2,-1               FILE7-DISPLAY FILE 50K; SPS --> GRF.     010760
1068.  850,0,0,0,0,0                 JOB RUN COMPLETE.                        010770
1069.  860,0,1200,1,0,0              MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.010780
```

```
1070.  861,1,0,0,4,-1             FILE1-LOAD MODULE 120K; SPS --> DSK.      010790
1071.  861,10,0,1,4,0             FILE1 DSK --> FMP.                         010800
1072.  862,1,0,0,4,-1             FILE2-PATCH AND GRID SETUP 100K;SPS-->DSK010810
1073.  862,10,0,1,4,-1            FILE2 DSK --> FMP.                         010820
1074.  860,60,10,0,2,-1           FMP PATCH AND GRID PROCESSING - 10 SEC.    010830
1075.  860,60,60,0,0,-1           FMP FLOW CODE PROCESSING - 60 SEC.         010840
1076.  869,10,0,0,160,-1          FILE9-RAW RESULTS FILE 5M; FMP -->DSK.     010850
1077.  869,1,0,1,160,0            FILE9 DSK --> SPS.                         010860
1078.  868,10,0,0,3,-1            FILE8-RESULTS FILE 90K; FMP --> DSK.       010870
1079.  868,1,0,1,3,-1             FILE8 DSK --> SPS.                         010880
1080.  860,61,300,30,1,-1         SPS POST-PROCESSING;200 SEC,60%LOAD.       010890
1081.  867,20,0,0,2,-1            FILE7-DISPLAY FILE 50K; SPS --> GRF.       010900
1082.  860,0,0,0,0,0              JOB RUN COMPLETE.                          010910
1083.  870,0,1200,1,0,0           MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.010920
1084.  871,1,0,0,4,-1             FILE1-LOAD MODULE 120K; SPS --> DSK.      010930
1085.  871,10,0,1,4,0             FILE1 DSK --> FMP.                         010940
1086.  872,20,0,1,4,-1            FILE2-PATCH AND GRID SETUP 100K;GRF-->SPS010950
1087.  870,61,360,35,2,-1         SPS PATCH AND GRID PROCESSING;240S,70%    010960
1088.  873,1,0,0,92,-1            FILE3-PATCH AND GRID TRANSFORMED; 3M,TO D010970
1089.  873,10,0,1,92,-1           FILE3 DSK --> FMP.                         010980
1090.  870,60,60,0,2,-1           FMP FLOW CODE PROCESSING - 60 SEC.         010990
1091.  878,10,0,0,3,-1            FILE8-RESULTS FILE 90K; FMP --> DSK.       011000
1092.  878,1,0,1,3,-1             FILE8 DSK --> SPS.                         011010
1093.  870,61,300,30,1,-1         SPS POST-PROCESSING;200 SEC,60%LOAD.       011020
1094.  877,20,0,0,2,-1            FILE7-DISPLAY FILE 50K; SPS --> GRF.       011030
1095.  870,0,0,0,0,0              JOB RUN COMPLETE.                          011040
1096.  880,0,1200,1,0,0           MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.011050
1097.  881,1,0,0,4,-1             FILE1-LOAD MODULE 120K; SPS --> DSK.      011060
1098.  881,10,0,1,4,0             FILE1 DSK --> FMP.                         011070
1099.  882,1,0,0,4,-1             FILE2-PATCH AND GRID 100K; SPS --> DSK.   011080
1100.  882,10,0,1,4,-1            FILE2 DSK --> FMP.                         011090
1101.  880,60,10,0,2,-1           FMP PATCH AND GRID PROCESSING- 10 SEC.     011100
1102.  880,60,60,0,0,-1           FMP FLOW CODE PROCESSING - 60 SEC.         011110
1103.  889,10,0,0,220,-1          FILE9-RESTART FILE 7M; FMP --> DSK.        011120
1104.  888,10,0,0,3,-1            FILE8-RESULTS FILE 90K; FMP --> DSK.       011130
1105.  888,1,0,1,3,-1             FILE8 DSK --> SPS.                         011140
1106.  880,61,300,30,1,-1         SPS POST-PROCESSING;200 SEC,60%LOAD.       011150
1107.  887,20,0,0,2,-1            FILE7-DISPLAY FILE 50K; SPS --> GRF.       011160
1108.  880,0,0,0,0,0              JOB RUN COMPLETE.                          011170
1109.  890,0,1200,1,0,0           MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.011180
1110.  890,61,3,35,0,-1           SPS COMPILATION OF SOURCE CODE-2 SEC.      011190
1111.  891,1,0,0,4,-1             FILE1-LOAD MODULE 120K; SPS --> DSK.      011200
1112.  891,10,0,1,4,0             FILE1 DSK --> FMP.                         011210
1113.  892,1,0,0,4,-1             FILE2-PATCH AND GRID SETUP 100K;SPS-->DSK011220
1114.  892,10,0,1,4,-1            FILE2 DSK --> FMP.                         011230
1115.  890,60,60,0,2,-1           FMP FLOW CODE PROCESSING - 60 SEC.         011240
1116.  899,10,0,0,160,0           FILE9-RAW RESULTS 5M; FMP --> DSK.         011250
1117.  898,10,0,0,3,-1            FILE8-RESULTS FILE 90K; FMP --> DSK.       011260
1118.  898,1,0,1,3,-1             FILE8 DSK --> SPS.                         011270
1119.  890,61,300,30,1,-1         SPS POST-PROCESSING;200 SEC,60%LOAD.       011280
1120.  897,20,0,0,2,-1            FILE7-DISPLAY FILE 50K; SPS --> GRF.       011290
1121.  890,0,0,0,0,0              JOB RUN COMPLETE.                          011300
1122.  900,0,1200,1,0,0           MODEL 3-INTERARRIVAL TIME MEAN OF 20 MIN.011310
1123.  901,1,0,0,4,-1             FILE1-LOAD MODULE 120K; SPS --> DSK.      011320
1124.  901,10,0,1,4,0             FILE1 DSK --> FMP.                         011330
1125.  902,20,0,1,4,-1            FILE2-PATCH AND GRID SETUP 100K;GRF-->SPS011340
1126.  900,61,360,35,2,-1         SPS PATCH AND GRID PROCESSING;240S,70%    011350
1127.  903,1,0,0,92,-1            FILE3-PATCH AND GRID TRANSFORMED; 3M,TO D011360
1128.  903,10,0,1,92,-1           FILE3 DSK --> FMP.                         011370
1129.  900,60,60,0,2,-1           FMP FLOW CODE PROCESSING - 60 SEC.         011380
1130.  908,10,0,0,3,-1            FILE8-RESULTS FILE 90K; FMP --> DSK.       011390
1131.  908,1,0,1,3,-1             FILE8 DSK --> SPS.                         011400
1132.  900,61,300,30,1,-1         SPS POST-PROCESSING;200 SEC,60%LOAD.       011410
```

```
1133.  907,20,0,0,2,-1              FILE7-DISPLAY FILE 50K: SPS --> GRF.       011420
1134.  900,0,0,0,0,0               JOB RUN COMPLETE.                          011430
1135.  920,0,900,2,1,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN   011440
1136.  921,1,0,0,4,-1              FILE1-LOAD MODULE 120K: SPS --> DSK.       011450
1137.  921,10,0,1,4,0              FILE1 DSK --> FMP.                         011460
1138.  922,20,0,1,4,-1             FILE2-PATCH AND GRID SETUP 100K:GRF-->SPS011470
1139.  920,61,360,35,2,-1          SPS PATCH AND GRID PROCESSING:240S,70%     011480
1140.  923,1,0,0,92,-1             FILE3-PATCH AND GRID TRANSFORMED: 3M,To D011490
1141.  923,10,0,1,92,-1            FILE3 DSK --> FMP.                         011500
1142.  920,60,600,0,2,-1            FMP FLOW CODE PROCESSING - 600 SEC.       011510
1143.  928,10,0,0,8,-1             FILE8-RESULTS FILE 225K: FMP --> DSK.      011520
1144.  928,1,0,1,8,-1              FILE8 DSK --> SPS.                         011530
1145.  920,61,360,30,1,-1          SPS POST-PROCESSING:240 SEC,60%LOAD.       011540
1146.  920,61,360,30,0,-1            "  "    "        "       "              011550
1147.  927,20,0,0,3,-1             FILE7-DISPLAY FILE 75K: SPS --> GRF.       011560
1148.  920,0,0,0,0,0               JOB RUN COMPLETE.                          011570
1149.  930,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN   011580
1150.  931,1,0,0,4,-1              FILE1-LOAD MODULE 120K: SPS --> DSK.       011590
1151.  931,10,0,1,4,0              FILE1 DSK --> FMP.                         011600
1152.  932,1,0,0,4,-1              FILE2-PATCH AND GRID 100K: SPS --> DSK.    011610
1153.  932,10,0,1,4,-1             FILE2 DSK --> FMP.                         011620
1154.  930,60,10,0,2,-1            FMP PATCH AND GRID PROCESSING- 10 SEC.     011630
1155.  930,60,600,0,0,-1            FMP FLOW CODE PROCESSING - 600 SEC.       011640
1156.  939,10,0,0,310,-1           FILE9-RESTART FILE 10M: FMP --> DSK.       011650
1157.  938,10,0,0,8,-1             FILE8-RESULTS FILE 225K: FMP --> DSK.      011660
1158.  938,1,0,1,8,-1              FILE8 DSK --> SPS.                         011670
1159.  930,61,360,30,1,-1          SPS POST-PROCESSING:240 SEC,60%LOAD.       011680
1160.  930,61,360,30,0,-1            "  "    "        "       "              011690
1161.  937,20,0,0,3,-1             FILE7-DISPLAY FILE 75K: SPS --> GRF.       011700
1162.  930,0,0,0,0,0               JOB RUN COMPLETE.                          011710
1163.  940,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN   011720
1164.  940,61,3,35,0,-1            SPS COMPILATION OF SOURCE CODE-2 SEC.      011730
1165.  941,1,0,0,4,-1              FILE1-LOAD MODULE 120K: SPS --> DSK.       011740
1166.  941,10,0,1,4,0              FILE1 DSK --> FMP.                         011750
1167.  942,1,0,0,4,-1              FILE2-PATCH AND GRID SETUP 100K:SPS-->DSK011760
1168.  942,10,0,1,4,-1             FILE2 DSK --> FMP.                         011770
1169.  940,60,600,0,2,-1            FMP FLOW CODE PROCESSING - 600 SEC.       011780
1170.  949,10,0,0,160,0            FILE9-RAW RESULTS 5M: FMP --> DSK.         011790
1171.  948,10,0,0,8,-1             FILE8-RESULTS FILE 225K: FMP --> DSK.      011800
1172.  948,1,0,1,8,-1              FILE8 DSK --> SPS.                         011810
1173.  940,61,360,30,1,-1          SPS POST-PROCESSING:240 SEC,60%LOAD.       011820
1174.  940,61,360,30,0,-1             "  "  "       "       "                011830
1175.  947,20,0,0,3,-1             FILE7-DISPLAY FILE 75K: SPS --> GRF.       011840
1176.  940,0,0,0,0,0               JOB RUN COMPLETE.                          011850
1177.  950,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN   011860
1178.  951,1,0,0,4,-1              FILE1-LOAD MODULE 120K: SPS --> DSK.       011870
1179.  951,10,0,1,4,0              FILE1 DSK --> FMP.                         011880
1180.  952,20,0,1,4,-1             FILE2-PATCH AND GRID SETUP 100K:GRF-->SPS011890
1181.  950,61,360,35,2,-1          SPS PATCH AND GRID PROCESSING-240S,70%     011900
1182.  953,1,0,0,46,-1             FILE3-RESULTING TRANSFORMED FILE SPS-->DS011910
1183.  953,10,0,1,46,0             FILE3 DSK --> FMP.                         011920
1184.  953,1,0,0,46,-1             REST OF FILE3- 3M TOTAL: SPS-->DSK.        011930
1185.  953,10,0,1,46,-1            REST OF FILE3 DSK --> FMP.                 011940
1186.  950,60,600,0,2,-1            FMP FLOW CODE PROCESSING - 600 SEC.       011950
1187.  958,10,0,0,8,-1             FILE8-RESULTS FILE 225K: FMP --> DSK.      011960
1188.  958,1,0,1,8,-1              FILE8 DSK --> SPS.                         011970
1189.  950,61,360,30,1,-1          SPS POST-PROCESSING:240 SEC,60%LOAD.       011980
1190.  950,61,360,30,0,-1            "  "    "        "       "              011990
1191.  957,20,0,0,3,-1             FILE7-DISPLAY FILE 75K: SPS --> GRF.       012000
1192.  950,0,0,0,0,0               JOB RUN COMPLETE.                          012010
1193.  960,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN   012020
1194.  961,1,0,0,4,-1              FILE1-LOAD MODULE 120K: SPS --> DSK.       012030
1195.  961,10,0,1,4,0              FILE1 DSK --> FMP.                         012040
```

```
1196.  962,1,0,0,4,-1              FILE2-PATCH AND GRID SETUP 100K;SPS-->DSK012050
1197.  962,10,0,1,4,-1             FILE2 DSK --> FMP.                       012060
1198.  960,60,10,0,2,-1            FMP PATCH AND GRID PROCESSING - 10 SEC.  012070
1199.  960,60,600,0,0,-1            FMP FLOW CODE PROCESSING - 600 SEC.     012080
1200.  969,10,0,0,160,-1           FILE9-RAW RESULTS FILE 5M; FMP -->DSK.   012090
1201.  969,1,0,1,160,0             FILE9 DSK --> SPS.                       012100
1202.  968,10,0,0,8,-1             FILE8-RESULTS FILE 225K; FMP --> DSK.    012110
1203.  968,1,0,1,8,-1              FILE8 DSK --> SPS.                       012120
1204.  960,61,360,30,1,-1          SPS POST-PROCESSING;240 SEC,60%LOAD.     012130
1205.  960,61,360,30,0,-1            "  "   "          "       "            012140
1206.  967,20,0,0,3,-1             FILE7-DISPLAY FILE 75K; SPS --> GRF.     012150
1207.  960,0,0,0,0,0               JOB RUN COMPLETE.                        012160
1208.  970,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN 012170
1209.  971,1,0,0,4,-1              FILE1-LOAD MODULE 120K; SPS --> DSK.     012180
1210.  971,10,0,1,4,0              FILE1 DSK --> FMP.                       012190
1211.  972,1,0,0,4,-1              FILE2-PATCH AND GRID 100K; SPS --> DSK.  012200
1212.  972,10,0,1,4,-1             FILE2 DSK --> FMP.                       012210
1213.  970,60,10,0,2,-1            FMP PATCH AND GRID PROCESSING- 10 SEC.   012220
1214.  970,60,600,0,0,-1            FMP FLOW CODE PROCESSING - 600 SEC.     012230
1215.  979,10,0,0,310,-1           FILE9-RESTART FILE 10M; FMP --> DSK.     012240
1216.  978,10,0,0,8,-1             FILE8-RESULTS FILE 225K; FMP --> DSK.    012250
1217.  978,1,0,1,8,-1              FILE8 DSK --> SPS.                       012260
1218.  970,61,360,30,1,-1          SPS POST-PROCESSING;240 SEC,60%LOAD.     012270
1219.  970,61,360,30,0,-1            "  "   "          "       "            012280
1220.  977,20,0,0,3,-1             FILE7-DISPLAY FILE 75K; SPS --> GRF.     012290
1221.  970,0,0,0,0,0               JOB RUN COMPLETE.                        012300
1222.  980,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN 012310
1223.  981,1,0,0,4,-1              FILE1-LOAD MODULE 120K; SPS --> DSK.     012320
1224.  981,10,0,1,4,0              FILE1 DSK --> FMP.                       012330
1225.  982,20,0,1,4,-1             FILE2-PATCH AND GRID SETUP 100K;GRF-->SPS012340
1226.  980,61,360,35,2,-1          SPS PATCH AND GRID PROCESSING-240S,70%.  012350
1227.  983,1,0,0,46,-1             FILE3-RESULTING TRANSFORMED FILE SPS-->DS012360
1228.  983,10,0,1,46,0             FILE3 DSK --> FMP.                       012370
1229.  983,1,0,0,46,-1             REST OF FILE3- 3M TOTAL; SPS-->DSK.      012380
1230.  983,10,0,1,46,-1            REST OF FILE3 DSK --> FMP.               012390
1231.  980,60,600,0,2,-1            FMP FLOW CODE PROCESSING - 600 SEC.     012400
1232.  988,10,0,0,8,-1             FILE8-RESULTS FILE 225K; FMP --> DSK.    012410
1233.  988,1,0,1,8,-1              FILE8 DSK --> SPS.                       012420
1234.  980,61,360,30,1,-1          SPS POST-PROCESSING;240 SEC,60%LOAD.     012430
1235.  980,61,360,30,0,-1            "  "   "          "       "            012440
1236.  987,20,0,0,3,-1             FILE7-DISPLAY FILE 75K; SPS --> GRF.     012450
1237.  980,0,0,0,0,0               JOB RUN COMPLETE.                        012460
1238.  990,0,900,1,0,0             MODEL 4-INTERARRIVAL TIME MEAN OF 15 MIN 012470
1239.  991,1,0,0,4,-1              FILE1-LOAD MODULE 120K; SPS --> DSK.     012480
1240.  991,10,0,1,4,0              FILE1 DSK --> FMP.                       012490
1241.  992,1,0,0,4,-1              FILE2-PATCH AND GRID 100K; SPS --> DSK.  012500
1242.  992,10,0,1,4,-1             FILE2 DSK --> FMP.                       012510
1243.  990,60,10,0,2,-1            FMP PATCH AND GRID PROCESSING- 10 SEC.   012520
1244.  990,60,600,0,0,-1            FMP FLOW CODE PROCESSING - 600 SEC.     012530
1245.  998,10,0,0,8,-1             FILE8-RESULTS FILE 225K; FMP --> DSK.    012540
1246.  998,1,0,1,8,-1              FILE8 DSK --> SPS.                       012550
1247.  990,61,360,30,1,-1          SPS POST-PROCESSING;240 SEC,60%LOAD.     012560
1248.  990,61,360,30,0,-1            "  "   "          "       "            012570
1249.  997,20,0,0,3,-1             FILE7-DISPLAY FILE 75K; SPS --> GRF.     012580
1250.  990,0,0,0,0,0               JOB RUN COMPLETE.                        012590
1251.  0,0,0,0,0,0                 END SIMULATION.                          012600
```

ARRIVAL TIME FOR INDIVIDUAL JOBS

JOB NO.	ASSIGNED SPS DEVICE	ARRIVAL TIME AT SPS EXECUTE QUEUE (SEC)
92	LEAD	0
1	BACKUP	157
2	BACKUP	251
3	BACKUP	354
4	LEAD	390
85	LEAD	396
5	BACKUP	745
6	BACKUP	775
61	LEAD	813
7	LEAD	940
8	BACKUP	1032
9	BACKUP	1087
93	BACKUP	1102
10	BACKUP	1106
94	BACKUP	1171
11	BACKUP	1297
62	BACKUP	1349
63	BACKUP	1355
12	BACKUP	1487
13	BACKUP	1686
14	LEAD	1801
15	LEAD	1816
64	LEAD	2014
16	LEAD	2039
17	LEAD	2051
18	LEAD	2148
19	LEAD	2223

20	LEAD	2262
21	LEAD	2646
22	LEAD	2831
65	LEAD	2882
23	LEAD	2958
66	LEAD	2965
24	LEAD	3031
67	LEAD	3077
25	LEAD	3208
95	LEAD	3227
26	BACKUP	3231
68	BACKUP	3280
69	BACKUP	3338
70	BACKUP	3530
86	BACKUP	3682
71	BACKUP	3729
27	BACKUP	3750
96	BACKUP	3754
72	LEAD	3799
97	LEAD	3926
28	LEAD	3932
73	LEAD	4152
29	LEAD	4532
87	LEAD	4576
88	BACKUP	4889
98	BACKUP	4900
30	LEAD	4992
31	LEAD	5035
32	LEAD	5038
33	LEAD	5102
34	LEAD	5257
35	LEAD	5366

36	LEAD	5518
37	LEAD	5523
38	LEAD	5590
39	LEAD	5656
40	LEAD	5690
74	LEAD	5879
41	LEAD	5880
99	BACKUP	5944
42	BACKUP	6078
43	BACKUP	6088
75	BACKUP	6148
76	BACKUP	6215
44	BACKUP	6357
77	BACKUP	6486
78	BACKUP	6522
45	BACKUP	6530
79	BACKUP	6767
46	LEAD	6926
47	LEAD	7089
48	LEAD	7148
80	LEAD	7302
49	LEAD	7347
50	LEAD	7354
89	LEAD	7357
51	BACKUP	7358
52	LEAD	7535
81	LEAD	7595

```
                                    SUMMARY OF JOB #    1

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  157  SEC., AND HAS SEIZED IT AT REL. CLK.   157 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  758  SEC., AND HAS SEIZED IT AT REL. CLK.   758 FOR     1  SECONDS.
    1              1              SPS      DSC               761                             0
    1              1              DSC      FMP               762                             0
    2              1              GPH2     DSC               764                             1
    2              1              DSC      FMP               765                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK.  766  SEC., AND HAS SEIZED IT AT REL. CLK.   999 FOR    10  SECONDS.
    8              2              FMP      DSC              1010                             0
    8              2              DSC      SPS              1012                             0
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1013  SEC., AND HAS SEIZED IT AT REL. CLK.  1013 FOR    60  SECONDS.
                     JOB #  1 COMPLETE AT RELATIVE CLOCK =  1074 SEC.


                                    SUMMARY OF JOB #    2

ILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  251  SEC., AND HAS SEIZED IT AT REL. CLK.   251 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  852  SEC., AND HAS SEIZED IT AT REL. CLK.   852 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1153  SEC., AND HAS SEIZED IT AT REL. CLK.  1153 FOR     1  SECONDS.
    1              1              SPS      DSC              1155                             0
    1              1              DSC      FMP              1156                             0
    2              1              GPH1     DSC              1157                             1
    2              1              DSC      FMP              1159                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1160  SEC., AND HAS SEIZED IT AT REL. CLK.  1719 FOR    10  SECONDS.
    8              2              FMP      DSC              1730                             0
    8              2              DSC      SPS              1732                             0
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1733  SEC., AND HAS SEIZED IT AT REL. CLK.  1733 FOR    60  SECONDS.
                     JOB #  2 COMPLETE AT RELATIVE CLOCK =  1794 SEC.
```

```
                              SUMMARY OF JOB #   3

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
        REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  354  SEC., AND HAS SEIZED IT AT REL. CLK.   354 FOR     1  SECONDS.
    1              1              SPS      DSC                356                        0
    1              1              DSC      FMP                357                        0
    2              1              GPH1     DSC                359                        1
    2              1              DSC      FMP                360                        0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK.  361  SEC., AND HAS SEIZED IT AT REL. CLK.   361 FOR    10  SECONDS.
    8              2              FMP      DSC                372                        0
    8              2              DSC      SPS                374                        0
        REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  375  SEC., AND HAS SEIZED IT AT REL. CLK.   375 FOR    60  SECONDS.
                  JOB #  3 COMPLETE AT RELATIVE CLOCK =   436 SEC.




                              SUMMARY OF JOB #   4

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  390  SEC., AND HAS SEIZED IT AT REL. CLK.   390 FOR     1  SECONDS.
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  991  SEC., AND HAS SEIZED IT AT REL. CLK.   991 FOR     1  SECONDS.
    1              1              SPS      DSC                993                        0
    1              1              DSC      FMP                994                        0
    2              1              GPH1     DSC                996                        1
    2              1              DSC      FMP                997                        0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK.  998  SEC., AND HAS SEIZED IT AT REL. CLK.  1079 FOR    10  SECONDS.
    9             90              FMP      DSC               1093                        3
    9             90              DSC      SPS               1170                       26
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1171  SEC., AND HAS SEIZED IT AT REL. CLK.  1171 FOR    60  SECONDS.
                  JOB #  4 COMPLETE AT RELATIVE CLOCK =  1232 SEC.
```

```
                                SUMMARY OF JOB #   5

FILE NO.       NUMBER OF           FROM      TO           TRANSFER COMPLETED AT              DURATION
          BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  745  SEC.. AND HAS SEIZED IT AT REL. CLK.   745 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1346  SEC.. AND HAS SEIZED IT AT REL. CLK.  1346 FOR     1  SECONDS.
   1                1                SPS     DSC                1349                               0
   1                1                DSC     FMP                1350                               0
   2                1               GPH2     DSC                1351                               1
   2                1                DSC     FMP                1352                               0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1353  SEC.. AND HAS SEIZED IT AT REL. CLK.  2339 FOR    10  SECONDS.
   8                2                FMP     DSC                2350                               0
   8                2                DSC     SPS                2352                               0
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2353  SEC.. AND HAS SEIZED IT AT REL. CLK.  2372 FOR    60  SECONDS.
                    JOB #  5 COMPLETE AT RELATIVE CLOCK =  2433 SEC.


                                SUMMARY OF JOB #   6

FILE NO.       NUMBER OF           FROM      TO           TRANSFER COMPLETED AT              DURATION
          BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  775  SEC.. AND HAS SEIZED IT AT REL. CLK.   775 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1376  SEC.. AND HAS SEIZED IT AT REL. CLK.  1376 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1677  SEC.. AND HAS SEIZED IT AT REL. CLK.  1677 FOR     1  SECONDS.
   1                1                SPS     DSC                1680                               0
   1                1                DSC     FMP                1681                               0
   2                1               GPH1     DSC                1682                               1
   2                1                DSC     FMP                1683                               0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1684  SEC.. AND HAS SEIZED IT AT REL. CLK.  2479 FOR    10  SECONDS.
   8                2                FMP     DSC                2490                               0
   8                2                DSC     SPS                2519                               0
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2520  SEC.. AND HAS SEIZED IT AT REL. CLK.  2580 FOR    60  SECONDS.
                    JOB #  6 COMPLETE AT RELATIVE CLOCK =  2641 SEC.
```

```
                               SUMMARY OF JOB #   7

FILE NO.       NUMBER OF           FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  940  SEC.. AND HAS SEIZED IT AT REL. CLK.   940 FOR     1  SECONDS.
    1             1                SPS       DSC                  942                            0
    1             1                DSC       FMP                  943                            0
    2             1               GPH2       DSC                  945                            1
    2             1                DSC       FMP                  946                            0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK.  947  SEC.. AND HAS SEIZED IT AT REL. CLK.  1069 FOR    10  SECONDS.
    8             2                FMP       DSC                 1080                            0
    8             2                DSC       SPS                 1082                            0
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1083  SEC.. AND HAS SEIZED IT AT REL. CLK.  1083 FOR    60  SECONDS.
                     JOB #  7 COMPLETE AT RELATIVE CLOCK =  1144 SEC.



                               SUMMARY OF JOB #   8

FILE NO.       NUMBER OF           FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1032  SEC.. AND HAS SEIZED IT AT REL. CLK.  1032 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1633  SEC.. AND HAS SEIZED IT AT REL. CLK.  1633 FOR     1  SECONDS.
    1             1                SPS       DSC                 1635                            0
    1             1                DSC       FMP                 1636                            0
    2             1               GPH2       DSC                 1637                            1
    2             1                DSC       FMP                 1639                            0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1640  SEC.. AND HAS SEIZED IT AT REL. CLK.  2469 FOR    10  SECONDS.
    9            90                FMP       DSC                 2486                            5
    9            90                DSC       SPS                 2518                           26
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2519  SEC.. AND HAS SEIZED IT AT REL. CLK.  2520 FOR    60  SECONDS.
                     JOB #  8 COMPLETE AT RELATIVE CLOCK =  2581 SEC.
```

```
                                   SUMMARY OF JOB #   9

FILE NO.         NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1087  SEC., AND HAS SEIZED IT AT REL. CLK.  1087 FOR     1  SECONDS.

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1688  SEC., AND HAS SEIZED IT AT REL. CLK.  1688 FOR     1  SECONDS.

    1               1                 SPS       DSC              1690                           0

    1               1                 DSC       FMP              1692                           0

    2               1                GPH2       DSC              1693                           1

    2               1                 DSC       FMP              1694                           0

           REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1695  SEC., AND HAS SEIZED IT AT REL. CLK.  2489 FOR    10  SECONDS.

    8               2                 FMP       DSC              2500                           0

    8               2                 DSC       SPS              2520                           0

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2521  SEC., AND HAS SEIZED IT AT REL. CLK.  2640 FOR    60  SECONDS.

                      JOB #  9 COMPLETE AT RELATIVE CLOCK =  2701 SEC.



                                   SUMMARY OF JOB #  10

FILE NO.         NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1106  SEC., AND HAS SEIZED IT AT REL. CLK.  1106 FOR     1  SECONDS.

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1707  SEC., AND HAS SEIZED IT AT REL. CLK.  1707 FOR     1  SECONDS.

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2008  SEC., AND HAS SEIZED IT AT REL. CLK.  2008 FOR     1  SECONDS.

    1               1                 SPS       DSC              2011                           0

    1               1                 DSC       FMP              2012                           0

    2               1                GPH1       DSC              2013                           1

    2               1                 DSC       FMP              2015                           0

           REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2016  SEC., AND HAS SEIZED IT AT REL. CLK.  2509 FOR    10  SECONDS.

    8               2                 FMP       DSC              2520                           0

    8               2                 DSC       SPS              2657                           1

           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2658  SEC., AND HAS SEIZED IT AT REL. CLK.  2661 FOR    60  SECONDS.

                      JOB # 10 COMPLETE AT RELATIVE CLOCK =  2722 SEC.
```

```
                                   SUMMARY OF JOB #  11

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
        BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1297  SEC., AND HAS SEIZED IT AT REL. CLK.  1297 FOR     1  SECONDS.
   1               1                SPS       DSC              1299                          0
   1               1                DSC       FMP              1300                          0
   2               1               GPH2       DSC              1301                          1
   2               1                DSC       FMP              1303                          0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1304  SEC., AND HAS SEIZED IT AT RFL. CLK.  2329 FOR    10  SECONDS.
   9              90                FMP       DSC              2343                          3
   9              90                DSC       SPS              2372                         27
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2373  SEC., AND HAS SEIZED IT AT RFL. CLK.  2432 FOR    60  SECONDS.
                   JOB # 11 COMPLETE AT RELATIVE CLOCK =  2493 SEC.



                                   SUMMARY OF JOB #  12

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
        BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1487  SEC., AND HAS SEIZED IT AT RFL. CLK.  1487 FOR     1  SECONDS.
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2088  SEC., AND HAS SEIZED IT AT RFL. CLK.  2088 FOR     1  SECONDS.
   1               1                SPS       DSC              2090                          0
   1               1                DSC       FMP              2091                          0
   2               1               GPH2       DSC              2093                          1
   2               1                DSC       FMP              2094                          0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2095  SEC., AND HAS SEIZED IT AT RFL. CLK.  2519 FOR    10  SECONDS.
   9              90                FMP       DSC              2533                          3
   9              90                DSC       SPS              2687                         31
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2688  SEC., AND HAS SEIZED IT AT RFL. CLK.  2688 FOR    60  SECONDS.
                   JOB # 12 COMPLETE AT RELATIVE CLOCK =  2749 SEC.
```

```
                                        SUMMARY OF JOB #  13

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
            BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1686  SEC., AND HAS SEIZED IT AT RFL. CLK.  1686 FOR     1  SECONDS.

            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2287  SEC., AND HAS SEIZED IT AT RFL. CLK.  2287 FOR     1  SECONDS.

      1              1              SPS      DSC              2289                            0

      1              1              DSC      FMP              2291                            0

      2              1             GPH1      DSC              2292                            1

      2              1              DSC      FMP              2293                            0

            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2294  SEC., AND HAS SEIZED IT AT RFL. CLK.  2599 FOR    10  SECONDS.

      8              2              FMP      DSC              2610                            0

      8              2              DSC      SPS              2661                            1

            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2662  SEC., AND HAS SEIZED IT AT RFL. CLK.  2664 FOR    60  SECONDS.

                         JOB # 13 COMPLETE AT RELATIVE CLOCK =  2725 SEC.
```

```
                                        SUMMARY OF JOB #  14

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
            BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1801  SEC., AND HAS SEIZED IT AT RFL. CLK.  1801 FOR     1  SECONDS.

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2402  SEC., AND HAS SEIZED IT AT RFL. CLK.  2402 FOR     1  SECONDS.

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2703  SEC., AND HAS SEIZED IT AT RFL. CLK.  2703 FOR     1  SECONDS.

      1              1              SPS      DSC              2706                            0

      1              1              DSC      FMP              2708                            0

      2              1             GPH2      DSC              2710                            2

      2              1              DSC      FMP              2711                            0

            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2712  SEC., AND HAS SEIZED IT AT RFL. CLK.  2721 FOR    10  SECONDS.

      8              2              FMP      DSC              2732                            0

      8              2              DSC      SPS              2734                            0

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2735  SEC., AND HAS SEIZED IT AT RFL. CLK.  2736 FOR    60  SECONDS.

                         JOB # 14 COMPLETE AT RELATIVE CLOCK =  2797 SEC.
```

```
                                   SUMMARY OF JOB #  15

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
           BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
           REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1816  SEC.. AND HAS SEIZED IT AT REL. CLK.  1816 FOR     1  SECONDS.
    1              1               SPS       DSC               1819                           0
    1              1               DSC       FMP               1820                           0
    2              1               GPH2      DSC               1821                           1
    2              1               DSC       FMP               1823                           0
           REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1824  SEC.. AND HAS SEIZED IT AT REL. CLK.  2499 FOR    10  SECONDS.
    8              2               FMP       DSC               2510                           0
    8              2               DSC       SPS               2512                           0
           REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2513  SEC.. AND HAS SEIZED IT AT REL. CLK.  2513 FOR    60  SECONDS.
                      JOB # 15 COMPLETE AT RELATIVE CLOCK =  2574 SEC.



                                   SUMMARY OF JOB #  16

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
           BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
           REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2039  SEC.. AND HAS SEIZED IT AT REL. CLK.  2039 FOR     1  SECONDS.
           REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2640  SEC.. AND HAS SEIZED IT AT REL. CLK.  2640 FOR     1  SECONDS.
    1              1               SPS       DSC               2642                           0
    1              1               DSC       FMP               2644                           0
    2              1               GPH1      DSC               2645                           1
    2              1               DSC       FMP               2646                           0
           REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2647  SEC.. AND HAS SEIZED IT AT REL. CLK.  2691 FOR    10  SECONDS.
    9             90               FMP       DSC               2705                           3
    9             90               DSC       SPS               2736                          30
           REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2737  SEC.. AND HAS SEIZED IT AT REL. CLK.  2776 FOR    60  SECONDS.
                      JOB # 16 COMPLETE AT RELATIVE CLOCK =  2837 SEC.
```

```
                                      SUMMARY OF JOB #  17

  FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2051  SEC.. AND HAS SEIZED IT AT REL. CLK.  2051 FOR     1  SECONDS.

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2652  SEC.. AND HAS SEIZED IT AT REL. CLK.  2652 FOR     1  SECONDS.

     1              1                SPS       DSC               2654                               0

     1              1                DSC       FMP               2656                               0

     2              1               GPH2       DSC               2658                               2

     2              1                DSC       FMP               2659                               0

            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2660  SEC.. AND HAS SEIZED IT AT REL. CLK.  2711 FOR    10  SECONDS.

     8              2                FMP       DSC               2722                               0

     8              2                DSC       SPS               2725                               1

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2726  SEC.. AND HAS SEIZED IT AT REL. CLK.  2726 FOR    60  SECONDS.

                         JOB # 17 COMPLETE AT RELATIVE CLOCK =  2787 SEC.



                                      SUMMARY OF JOB #  18

  FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2148  SEC.. AND HAS SEIZED IT AT REL. CLK.  2148 FOR     1  SECONDS.

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2749  SEC.. AND HAS SEIZED IT AT REL. CLK.  2786 FOR     1  SECONDS.

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3087  SEC.. AND HAS SEIZED IT AT REL. CLK.  3087 FOR     1  SECONDS.

     1              1                SPS       DSC               3089                               0

     1              1                DSC       FMP               3090                               0

     2              1               GPH2       DSC               3091                               1

     2              1                DSC       FMP               3092                               0

            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3093  SEC.. AND HAS SEIZED IT AT REL. CLK.  3093 FOR    10  SECONDS.

     8              2                FMP       DSC               3105                               0

     8              2                DSC       SPS               3106                               0

            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3107  SEC.. AND HAS SEIZED IT AT REL. CLK.  3107 FOR    60  SECONDS.

                         JOB # 18 COMPLETE AT RELATIVE CLOCK =  3168 SEC.
```

```
                                   SUMMARY OF JOB #  19

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2223  SEC.. AND HAS SEIZED IT AT REL. CLK.  2223 FOR     1  SECONDS.
   1               1              SPS       DSC               2225                         0
   1               1              DSC       FMP               2226                         0
   2               1              GPH2      DSC               2228                         1
   2               1              DSC       FMP               2229                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2230  SEC.. AND HAS SEIZED IT AT REL. CLK.  2529 FOR    10  SECONDS.
   9              90              FMP       DSC               2543                         3
   9              90              DSC       SPS               2571                        26
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2572  SEC.. AND HAS SEIZED IT AT REL. CLK.  2572 FOR    60  SECONDS.
                    JOB # 19 COMPLETE AT RELATIVE CLOCK =  2633 SEC.




                                   SUMMARY OF JOB #  20

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2262  SEC.. AND HAS SEIZED IT AT REL. CLK.  2262 FOR     1  SECONDS.
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2863  SEC.. AND HAS SEIZED IT AT REL. CLK.  2863 FOR     1  SECONDS.
   1               1              SPS       DSC               2866                         0
   1               1              DSC       FMP               2867                         0
   2               1              GPH1      DSC               2868                         1
   2               1              DSC       FMP               2870                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2871  SEC.. AND HAS SEIZED IT AT REL. CLK.  2871 FOR    10  SECONDS.
   9              90              FMP       DSC               2885                         3
   9              90              DSC       SPS               2912                        26
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2913  SEC.. AND HAS SEIZED IT AT REL. CLK.  2913 FOR    60  SECONDS.
                    JOB # 20 COMPLETE AT RELATIVE CLOCK =  2974 SEC.
```

```
                              SUMMARY OF JOB #  21

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
        BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2646  SEC., AND HAS SEIZED IT AT REL. CLK.  2646 FOR     1  SECONDS.
    1               1              SPS      DSC                2648                              0
    1               1              DSC      FMP                2649                              0
    2               1             GPH1      DSC                2650                              1
    2               1              DSC      FMP                2652                              0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2653  SEC., AND HAS SEIZED IT AT REL. CLK.  2701 FOR    10  SECONDS.
    8               2              FMP      DSC                2712                              0
    8               2              DSC      SPS                2715                              1
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2716  SEC., AND HAS SEIZED IT AT REL. CLK.  2716 FOR    60  SECONDS.
                      JOB # 21 COMPLETE AT RELATIVE CLOCK =  2777 SEC.
```

```
                              SUMMARY OF JOB #  22

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
        BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2831  SEC., AND HAS SEIZED IT AT REL. CLK.  2831 FOR     1  SECONDS.
    1               1              SPS      DSC                2833                              0
    1               1              DSC      FMP                2834                              0
    2               1             GPH1      DSC                2835                              1
    2               1              DSC      FMP                2837                              0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2838  SEC., AND HAS SEIZED IT AT REL. CLK.  2838 FOR    10  SECONDS.
    8               2              FMP      DSC                2849                              0
    8               2              DSC      SPS                2850                              0
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2851  SEC., AND HAS SEIZED IT AT REL. CLK.  2851 FOR    60  SECONDS.
                      JOB # 22 COMPLETE AT RELATIVE CLOCK =  2912 SEC.
```

```
                                   SUMMARY OF JOB #  23

FILE NO.      NUMBER OF          FROM      TO           TRANSFER COMPLETED AT            DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2958  SEC.. AND HAS SEIZED IT AT REL. CLK.  2958 FOR     1  SECONDS.
    1              1               SPS      DSC               2961                             0
    1              1               DSC      FMP               2962                             0
    2              1              GPH1      DSC               2963                             1
    2              1               DSC      FMP               2964                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2965  SEC.. AND HAS SEIZED IT AT REL. CLK.  2965 FOR    10  SECONDS.
    8              2               FMP      DSC               2977                             0
    8              2               DSC      SPS               2978                             0
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2979  SEC.. AND HAS SEIZED IT AT REL. CLK.  2979 FOR    60  SECONDS.
                    JOB # 23 COMPLETE AT RELATIVE CLOCK =  3040 SEC.



                                   SUMMARY OF JOB #  24

FILE NO.      NUMBER OF          FROM      TO           TRANSFER COMPLETED AT            DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3031  SEC.. AND HAS SEIZED IT AT REL. CLK.  3031 FOR     1  SECONDS.
    1              1               SPS      DSC               3034                             0
    1              1               DSC      FMP               3035                             0
    2              1              GPH2      DSC               3036                             1
    2              1               DSC      FMP               3037                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3038  SEC.. AND HAS SEIZED IT AT REL. CLK.  3038 FOR    10  SECONDS.
    9             90               FMP      DSC               3053                             3
    9             90               DSC      SPS               3081                            26
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3082  SEC.. AND HAS SEIZED IT AT REL. CLK.  3082 FOR    60  SECONDS.
                    JOB # 24 COMPLETE AT RELATIVE CLOCK =  3143 SEC.
```

```
                              SUMMARY OF JOB #  25

FILE NO.        NUMBER OF          FROM      TO           TRANSFER COMPLETED AT              DURATION
          BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...    OF TRANSFER (SEC)
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3208  SEC.. AND HAS SEIZED IT AT REL. CLK.  3208 FOR     1  SECONDS.
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3809  SEC.. AND HAS SEIZED IT AT REL. CLK.  3809 FOR     1  SECONDS.
    1              1               SPS       DSC               3811                             0
    1              1               DSC       FMP               3812                             0
    2              1               GPH1      DSC               3814                             1
    2              1               DSC       FMP               3815                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3816  SEC.. AND HAS SEIZED IT AT REL. CLK.  4305 FOR    10  SECONDS.
    8              2               FMP       DSC               4317                             0
    8              2               DSC       SPS               4318                             0
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4319  SEC.. AND HAS SEIZED IT AT REL. CLK.  4319 FOR    60  SECONDS.
                    JOB # 25 COMPLETE AT RELATIVE CLOCK =  4380 SEC.


                              SUMMARY OF JOB #  26

FILE NO.        NUMBER OF          FROM      TO           TRANSFER COMPLETED AT              DURATION
          BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...    OF TRANSFER (SEC)
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3231  SEC.. AND HAS SEIZED IT AT REL. CLK.  3231 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3832  SEC.. AND HAS SEIZED IT AT REL. CLK.  3832 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4133  SEC.. AND HAS SEIZED IT AT REL. CLK.  4133 FOR     1  SECONDS.
    1              1               SPS       DSC               4135                             0
    1              1               DSC       FMP               4137                             0
    2              1               GPH2      DSC               4138                             1
    2              1               DSC       FMP               4139                             0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4140  SEC.. AND HAS SEIZED IT AT REL. CLK.  4445 FOR    10  SECONDS.
    8              2               FMP       DSC               4457                             0
    8              2               DSC       SPS               4458                             0
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4459  SEC.. AND HAS SEIZED IT AT REL. CLK.  4459 FOR    60  SECONDS.
                    JOB # 26 COMPLETE AT RELATIVE CLOCK =  4520 SEC.
```

```
                              SUMMARY OF JOB #  27

FILE NO.     NUMBER OF          FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3750  SEC., AND HAS SEIZED IT AT REL. CLK.  3750 FOR     1  SECONDS.
   1            1               SPS      DSC              3752                         0
   1            1               DSC      FMP              3754                         0
   2            1               GPH1     DSC              3756                         2
   2            1               DSC      FMP              3757                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3758  SEC., AND HAS SEIZED IT AT REL. CLK.  4285 FOR    10  SECONDS.
   8            2               FMP      DSC              4297                         0
   8            2               DSC      SPS              4298                         0
         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4299  SEC., AND HAS SEIZED IT AT REL. CLK.  4299 FOR    60  SECONDS.
                     JOB # 27 COMPLETE AT RELATIVE CLOCK =  4360 SEC.
```

```
                              SUMMARY OF JOB #  28

FILE NO.     NUMBER OF          FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3932  SEC., AND HAS SEIZED IT AT REL. CLK.  3932 FOR     1  SECONDS.
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4533  SEC., AND HAS SEIZED IT AT REL. CLK.  4533 FOR     1  SECONDS.
   1            1               SPS      DSC              4536                         0
   1            1               DSC      FMP              4537                         0
   2            1               GPH1     DSC              4538                         1
   2            1               DSC      FMP              4540                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4541  SEC., AND HAS SEIZED IT AT REL. CLK.  5895 FOR    10  SECONDS.
   9           90               FMP      DSC              5913                         6
   9           90               DSC      SPS              5942                        28
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5943  SEC., AND HAS SEIZED IT AT REL. CLK.  5943 FOR    60  SECONDS.
                     JOB # 28 COMPLETE AT RELATIVE CLOCK =  6004 SEC.
```

```
                                   SUMMARY OF JOB #  29

FILE NO.     NUMBER OF          FROM      TO           TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4532  SEC., AND HAS SEIZED IT AT RFL. CLK.  4532 FOR      1  SECONDS.

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5133  SEC., AND HAS SEIZED IT AT RFL. CLK.  5133 FOR      1  SECONDS.

   1             1              SPS       DSC              5136                        0

   1             1              DSC       FMP              5137                        0

   2             1              GPH2      DSC              5138                        1

   2             1              DSC       FMP              5139                        0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5140  SEC., AND HAS SEIZED IT AT RFL. CLK.  5985 FOR     10  SECONDS.

   8             2              FMP       DSC              5997                        0

   8             2              DSC       SPS              6004                        0

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6005  SEC., AND HAS SEIZED IT AT RFL. CLK.  6005 FOR     60  SECONDS.

                JOB # 29 COMPLETE AT RELATIVE CLOCK =  6066 SEC.



                                   SUMMARY OF JOB #  30

FILE NO.     NUMBER OF          FROM      TO           TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4992  SEC., AND HAS SEIZED IT AT RFL. CLK.  4992 FOR      1  SECONDS.

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5593  SEC., AND HAS SEIZED IT AT REL. CLK.  5593 FOR      1  SECONDS.

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5894  SEC., AND HAS SEIZED IT AT RFL. CLK.  5894 FOR      1  SECONDS.

   1             1              SPS       DSC              5897                        0

   1             1              DSC       FMP              5898                        0

   2             1              GPH1      DSC              5899                        1

   2             1              DSC       FMP              5901                        0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5902  SEC., AND HAS SEIZED IT AT RFL. CLK.  6705 FOR     10  SECONDS.

   9            90              FMP       DSC              6723                        6

   9            90              DSC       SPS              6765                       40

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6766  SEC., AND HAS SEIZED IT AT RFL. CLK.  6766 FOR     60  SECONDS.

                JOB # 30 COMPLETE AT RELATIVE CLOCK =  6827 SEC.
```

```
                              SUMMARY OF JOB #  31

FILE NO.       NUMBER OF          FROM     TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5035  SEC., AND HAS SEIZED IT AT REL. CLK.  5035 FOR     1  SECONDS.
    1              1              SPS     DSC               5037                               0
    1              1              DSC     FMP               5038                               0
    2              1              GPH2    DSC               5040                               1
    2              1              DSC     FMP               5041                               0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5042  SEC., AND HAS SEIZED IT AT REL. CLK.  5975 FOR    10  SECONDS.
    8              2              FMP     DSC               5987                               0
    8              2              DSC     SPS               6004                               1
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6005  SEC., AND HAS SEIZED IT AT REL. CLK.  6065 FOR    60  SECONDS.
                      JOB # 31 COMPLETE AT RELATIVE CLOCK =  6126 SEC.
```

```
                              SUMMARY OF JOB #  32

FILE NO.       NUMBER OF          FROM     TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5038  SEC., AND HAS SEIZED IT AT REL. CLK.  5038 FOR     1  SECONDS.
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5639  SEC., AND HAS SEIZED IT AT REL. CLK.  5639 FOR     1  SECONDS.
    1              1              SPS     DSC               5641                               0
    1              1              DSC     FMP               5643                               0
    2              1              GPH1    DSC               5644                               1
    2              1              DSC     FMP               5645                               0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5646  SEC., AND HAS SEIZED IT AT REL. CLK.  6665 FOR    10  SECONDS.
    9             90              FMP     DSC               6680                               3
    9             90              DSC     SPS               6730                              49
          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6731  SEC., AND HAS SEIZED IT AT REL. CLK.  6731 FOR    60  SECONDS.
                      JOB # 32 COMPLETE AT RELATIVE CLOCK =  6792 SEC.
```

```
                                    SUMMARY OF JOB #  33

FILE NO.        NUMBER OF            FROM      TO            TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5102  SEC., AND HAS SEIZED IT AT REL. CLK.  5102 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5703  SEC., AND HAS SEIZED IT AT REL. CLK.  5703 FOR     1  SECONDS.

    1                 1                SPS      DSC                5705                           0

    1                 1                DSC      FMP                5706                           0

    2                 1               GPH1      DSC                5708                           1

    2                 1                DSC      FMP                5709                           0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5710  SEC., AND HAS SEIZED IT AT REL. CLK.  6685 FOR    10  SECONDS.

    8                 2                FMP      DSC                6697                           0

    8                 2                DSC      SPS                6700                           2

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6701  SEC., AND HAS SEIZED IT AT REL. CLK.  6701 FOR    60  SECONDS.
                      JOB # 33 COMPLETE AT RELATIVE CLOCK =  6762 SEC.


                                    SUMMARY OF JOB #  34

FILE NO.        NUMBER OF            FROM      TO            TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5257  SEC., AND HAS SEIZED IT AT REL. CLK.  5257 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5858  SEC., AND HAS SEIZED IT AT REL. CLK.  5858 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6159  SEC., AND HAS SEIZED IT AT REL. CLK.  6159 FOR     1  SECONDS.

    1                 1                SPS      DSC                6161                           0

    1                 1                DSC      FMP                6163                           0

    2                 1               GPH2      DSC                6164                           1

    2                 1                DSC      FMP                6165                           0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6166  SEC., AND HAS SEIZED IT AT REL. CLK.  6825 FOR    10  SECONDS.

    8                 2                FMP      DSC                6837                           0

    8                 2                DSC      SPS                6839                           1

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6840  SEC., AND HAS SEIZED IT AT REL. CLK.  6840 FOR    60  SECONDS.
                      JOB # 34 COMPLETE AT RELATIVE CLOCK =  6901 SEC.
```

```
                              SUMMARY OF JOB #  35

FILE NO.        NUMBER OF         FROM      TO          TRANSFER COMPLETED AT          DURATION
            BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5366  SEC.. AND HAS SEIZED IT AT REL. CLK.  5366 FOR     1  SECONDS.
    1              1              SPS.      DSC              5368                         0
    1              1              DSC       FMP              5369                         0
    2              1              GPH1      DSC              5373                         4
    2              1              DSC       FMP              5375                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5376  SEC.. AND HAS SEIZED IT AT REL. CLK.  6655 FOR    10  SECONDS.
    8              2              FMP       DSC              6667                         0
    8              2              DSC       SPS              6668                         0
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6669  SEC.. AND HAS SEIZED IT AT REL. CLK.  6669 FOR    60  SECONDS.
                        JOB # 35 COMPLETE AT RELATIVE CLOCK =  6730 SEC.
```

```
                              SUMMARY OF JOB #  36

FILE NO.        NUMBER OF         FROM      TO          TRANSFER COMPLETED AT          DURATION
            BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5518  SEC.. AND HAS SEIZED IT AT REL. CLK.  5518 FOR     1  SECONDS.
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6119  SEC.. AND HAS SEIZED IT AT REL. CLK.  6127 FOR     1  SECONDS.
    1              1              SPS       DSC              6129                         0
    1              1              DSC       FMP              6131                         0
    2              1              GPH2      DSC              6132                         1
    2              1              DSC       FMP              6133                         0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6134  SEC.. AND HAS SEIZED IT AT REL. CLK.  6805 FOR    10  SECONDS.
    9             90              FMP       DSC              6820                         3
    9             90              DSC       SPS              6849                        28
         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6850  SEC.. AND HAS SEIZED IT AT REL. CLK.  6850 FOR    60  SECONDS.
                        JOB # 36 COMPLETE AT RELATIVE CLOCK =  6911 SEC.
```

```
                                   SUMMARY OF JOB #  37

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
           BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5523  SEC., AND HAS SEIZED IT AT REL. CLK.  5523 FOR     1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6124  SEC., AND HAS SEIZED IT AT REL. CLK.  6128 FOR     1  SECONDS.

    1               1              SPS       DSC               6130                         0

    1               1              DSC       FMP               6132                         0

    2               1              GPH1      DSC               6133                         1

    2               1              DSC       FMP               6134                         0

          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6135  SEC., AND HAS SEIZED IT AT REL. CLK.  6815 FOR    10  SECONDS.

    8               2              FMP       DSC               6827                         0

    8               2              DSC       SPS               6829                         1

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6830  SEC., AND HAS SEIZED IT AT REL. CLK.  6830 FOR    60  SECONDS.
                    JOB # 37 COMPLETE AT RELATIVE CLOCK =  6891 SEC.




                                   SUMMARY OF JOB #  38

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
           BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5590  SEC., AND HAS SEIZED IT AT REL. CLK.  5590 FOR     1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6191  SEC., AND HAS SEIZED IT AT REL. CLK.  6191 FOR     1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6492  SEC., AND HAS SEIZED IT AT REL. CLK.  6492 FOR     1  SECONDS.

    1               1              SPS       DSC               6494                         0

    1               1              DSC       FMP               6495                         0

    2               1              GPH1      DSC               6497                         1

    2               1              DSC       FMP               6498                         0

          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6499  SEC., AND HAS SEIZED IT AT REL. CLK.  6915 FOR    10  SECONDS.

    8               2              FMP       DSC               6927                         0

    8               2              DSC       SPS               6928                         0

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6929  SEC., AND HAS SEIZED IT AT REL. CLK.  6929 FOR    60  SECONDS.
                    JOB # 38 COMPLETE AT RELATIVE CLOCK =  6990 SEC.
```

```
                                   SUMMARY OF JOB #  39

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5656  SEC.. AND HAS SEIZED IT AT REL. CLK.  5656 FOR     1  SECONDS.
    1              1               SPS      DSC               5658                                0
    1              1               DSC      FMP               5659                                0
    2              1               GPH1     DSC               5661                                1
    2              1               DSC      FMP               5662                                0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5663  SEC.. AND HAS SEIZED IT AT REL. CLK.  6675 FOR    10  SECONDS.
    9             90               FMP      DSC               6691                                4
    9             90               DSC      SPS               6749                               57
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6750  SEC.. AND HAS SEIZED IT AT REL. CLK.  6750 FOR    60  SECONDS.
                      JOB # 39 COMPLETE AT RELATIVE CLOCK =  6811 SEC.



                                   SUMMARY OF JOB #  40

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5690  SEC.. AND HAS SEIZED IT AT REL. CLK.  5690 FOR     1  SECONDS.
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6291  SEC.. AND HAS SEIZED IT AT REL. CLK.  6291 FOR     1  SECONDS.
    1              1               SPS      DSC               6294                                0
    1              1               DSC      FMP               6295                                0
    2              1               GPH2     DSC               6296                                1
    2              1               DSC      FMP               6297                                0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6298  SEC.. AND HAS SEIZED IT AT REL. CLK.  6835 FOR    10  SECONDS.
    9             90               FMP      DSC               6850                                4
    9             90               DSC      SPS               6896                               44
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6897  SEC.. AND HAS SEIZED IT AT REL. CLK.  6897 FOR    60  SECONDS.
                      JOB # 40 COMPLETE AT RELATIVE CLOCK =  6958 SEC.
```

```
                              SUMMARY OF JOB #  41

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
           BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5880  SEC., AND HAS SEIZED IT AT REL. CLK.  5880 FOR     1  SECONDS.
   1              1               SPS       DSC              5882                          0
   1              1               DSC       FMP              5883                          0
   2              1               GPH2      DSC              5884                          1
   2              1               DSC       FMP              5886                          0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5887  SEC., AND HAS SEIZED IT AT REL. CLK.  6695 FOR    10  SECONDS.
   8              2               FMP       DSC              6707                          0
   8              2               DSC       SPS              6709                          1
       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6710  SEC., AND HAS SEIZED IT AT REL. CLK.  6710 FOR    60  SECONDS.
                    JOB # 41 COMPLETE AT RELATIVE CLOCK =  6771 SEC.
```

```
                              SUMMARY OF JOB #  42

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT            DURATION
           BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6078  SEC., AND HAS SEIZED IT AT REL. CLK.  6078 FOR     1  SECONDS.
   1              1               SPS       DSC              6080                          0
   1              1               DSC       FMP              6082                          0
   2              1               GPH1      DSC              6083                          1
   2              1               DSC       FMP              6084                          0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6085  SEC., AND HAS SEIZED IT AT REL. CLK.  6785 FOR    10  SECONDS.
   8              2               FMP       DSC              6797                          0
   8              2               DSC       SPS              6798                          0
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6799  SEC., AND HAS SEIZED IT AT REL. CLK.  6799 FOR    60  SECONDS.
                    JOB # 42 COMPLETE AT RELATIVE CLOCK =  6860 SEC.
```

```
                              SUMMARY OF JOB #  43

FILE NO.        NUMBER OF           FROM      TO          TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6088  SEC., AND HAS SEIZED IT AT REL. CLK.  6088 FOR     1  SECONDS.
     1             1                 SPS      DSC               6090                               0
     1             1                 DSC      FMP               6092                               0
     2             1                 GPH2     DSC               6093                               1
     2             1                 DSC      FMP               6094                               0
            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6095  SEC., AND HAS SEIZED IT AT RFL. CLK.  6795 FOR    10  SECONDS.
     9            90                 FMP      DSC               6810                               3
     9            90                 DSC      SPS               6913                              53
            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6914  SEC., AND HAS SEIZED IT AT RFL. CLK.  6915 FOR    60  SECONDS.
                        JOB # 43 COMPLETE AT RELATIVE CLOCK =  6976 SEC.



                              SUMMARY OF JOB #  44

FILE NO.        NUMBER OF           FROM      TO          TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6357  SEC., AND HAS SEIZED IT AT RFL. CLK.  6357 FOR     1  SECONDS.
     1             1                 SPS      DSC               6359                               0
     1             1                 DSC      FMP               6361                               0
     2             1                 GPH1     DSC               6362                               1
     2             1                 DSC      FMP               6363                               0
            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6364  SEC., AND HAS SEIZED IT AT RFL. CLK.  6845 FOR    10  SECONDS.
     9            90                 FMP      DSC               6862                               5
     9            90                 DSC      SPS               6915                              52
            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6916  SEC., AND HAS SEIZED IT AT RFL. CLK.  6975 FOR    60  SECONDS.
                        JOB # 44 COMPLETE AT RELATIVE CLOCK =  7036 SEC.
```

```
                                   SUMMARY OF JOB #  45

FILE NO.        NUMBER OF          FROM      TO           TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6530  SEC., AND HAS SEIZED IT AT REL. CLK.  6530 FOR     1  SECONDS.
            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7131  SEC., AND HAS SEIZED IT AT REL. CLK.  7131 FOR     1  SECONDS.
     1               1              SPS      DSC               7133                            0
     1               1              DSC      FMP               7134                            0
     2               1              GPH1     DSC               7136                            1
     2               1              DSC      FMP               7137                            0
            REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7138  SEC., AND HAS SEIZED IT AT REL. CLK.  7655 FOR    10  SECONDS.
     8               2              FMP      DSC               7667                            0
     8               2              DSC      SPS               7670                            2
            REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7671  SEC., AND HAS SEIZED IT AT REL. CLK.  7671 FOR    60  SECONDS.



                                   SUMMARY OF JOB #  46

FILE NO.        NUMBER OF          FROM      TO           TRANSFER COMPLETED AT            DURATION
            BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6926  SEC., AND HAS SEIZED IT AT REL. CLK.  6926 FOR     1  SECONDS.
            REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7527  SEC., AND HAS SEIZED IT AT REL. CLK.  7527 FOR     1  SECONDS.
```

```
                          SUMMARY OF JOB #  47

FILE NO.     NUMBER OF          FROM     TO         TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

      REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7089  SEC., AND HAS SEIZED IT AT REL. CLK.  7089 FOR     1  SECONDS.

    1             1             SPS     DSC              7092                          0

    1             1             DSC     FMP              7093                          0

    2             1             GPH1    DSC              7094                          1

    2             1             DSC     FMP              7095                          0

      REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7096  SEC., AND HAS SEIZED IT AT REL. CLK.  7645 FOR    10  SECONDS.

    8             2             FMP     DSC              7657                          0

    8             2             DSC     SPS              7659                          0

      REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7660  SEC., AND HAS SEIZED IT AT REL. CLK.  7660 FOR    60  SECONDS.



                          SUMMARY OF JOB #  48

FILE NO.     NUMBER OF          FROM     TO         TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

      REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7148  SEC., AND HAS SEIZED IT AT REL. CLK.  7148 FOR     1  SECONDS.



                          SUMMARY OF JOB #  49

FILE NO.     NUMBER OF          FROM     TO         TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

      REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7347  SEC., AND HAS SEIZED IT AT REL. CLK.  7347 FOR     1  SECONDS.
```

```
                                  SUMMARY OF JOB #  50

FILE NO.       NUMBER OF            FROM      TO           TRANSFER COMPLETED AT            DURATION
        BLOCKS TRANSFERRED                               RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7354  SEC., AND HAS SEIZED IT AT REL. CLK.  7354 FOR     1  SECONDS.



                                  SUMMARY OF JOB #  51

FILE NO.       NUMBER OF            FROM      TO           TRANSFER COMPLETED AT            DURATION
        BLOCKS TRANSFERRED                               RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7358  SEC., AND HAS SEIZED IT AT REL. CLK.  7358 FOR     1  SECONDS.

   1              1                  SPS      DSC             7360                          0

   1              1                  DSC      FMP             7361                          0

   2              1                 GPH1      DSC             7363                          1

   2              1                  DSC      FMP             7364                          0



                                  SUMMARY OF JOB #  52

FILE NO.       NUMBER OF            FROM      TO           TRANSFER COMPLETED AT            DURATION
        BLOCKS TRANSFERRED                               RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7535  SEC., AND HAS SEIZED IT AT REL. CLK.  7535 FOR     1  SECONDS.

   1              1                  SPS      DSC             7537                          0

   1              1                  DSC      FMP             7538                          0

   2              1                 GPH1      DSC             7540                          1

   2              1                  DSC      FMP             7541                          0
```

```
                                   SUMMARY OF JOB #  61

FILE NO.      NUMBER OF          FROM      TO           TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  813  SEC., AND HAS SEIZED IT AT REL. CLK.   813 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1414  SEC., AND HAS SEIZED IT AT REL. CLK.  1414 FOR     1  SECONDS.

    1              2             SPS       DSC             1417                              0

    1              2             DSC       FMP             1418                              0

    2              2             GPH2      DSC             1421                              2

    2              2             DSC       FMP             1422                              0

    3             19             SPS       DSC             1427                              5

    3             19             DSC       FMP             1429                              0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1430  SEC., AND HAS SEIZED IT AT REL. CLK.  2349 FOR    60  SECONDS.

    9            125             FMP       DSC             2415                              4

    9            125             FMP       DSC             2421                              5

    9            125             DSC       SPS             2485                             69

    9            125             DSC       SPS             2490                             67

                  JOB # 61 COMPLETE AT RELATIVE CLOCK =  2491 SEC.
```

```
                                        SUMMARY OF JOB #  62

    FILE NO.        NUMBER OF           FROM      TO          TRANSFER COMPLETED AT          DURATION
               BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
              REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1349  SEC., AND HAS SEIZED IT AT REL. CLK.  1349 FOR     1  SECONDS.
       1                2                SPS      DSC                1351                          0
       1                2                DSC      FMP                1353                          0
       2                2                GPH1     SPS                1354                          1
              REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1355  SEC., AND HAS SEIZED IT AT REL. CLK.  1355 FOR    80  SECONDS.
       2                2                SPS      DSC                1436                          0
       2                2                DSC      FMP                1438                          0
       3               19                SPS      DSC                1443                          5
       3               19                DSC      FMP                1445                          0
              REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1446  SEC., AND HAS SEIZED IT AT REL. CLK.  2409 FOR    60  SECONDS.
       8                6                FMP      DSC                2471                          0
       8                6                DSC      SPS                2475                          2
              REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2476  SEC., AND HAS SEIZED IT AT REL. CLK.  2476 FOR   180  SECONDS.
       7                4                SPS     GPH1                2660                          2
                           JOB # 62 COMPLETE AT RELATIVE CLOCK =  2661 SEC.
```

```
                              SUMMARY OF JOB #  63

FILE NO.     NUMBER OF           FROM      TO          TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1355  SEC., AND HAS SEIZED IT AT REL. CLK.  1355 FOR     1  SECONDS.

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1956  SEC., AND HAS SEIZED IT AT REL. CLK.  1956 FOR     1  SECONDS.

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2257  SEC., AND HAS SEIZED IT AT REL. CLK.  2257 FOR     1  SECONDS.

    1              2              SPS     DSC              2260                       0

    1              2              DSC     FMP              2261                       0

    2              2             GPH2     DSC              2264                       3

    2              2              DSC     FMP              2265                       0

    3             19              SPS     DSC              2271                       5

    3             19              DSC     FMP              2272                       0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2273  SEC., AND HAS SEIZED IT AT REL. CLK.  2539 FOR    60  SECONDS.

    8             12              FMP     DSC              2601                       0

    8             12              DSC     SPS              2664                       8

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2665  SEC., AND HAS SEIZED IT AT REL. CLK.  2724 FOR   360  SECONDS.

    7              8              SPS    GPH2              3090                       4

                   JOB # 63 COMPLETE AT RELATIVE CLOCK =  3091 SEC.
```

```
                                     SUMMARY OF JOB #  64

FILE NO.        NUMBER OF         FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2014  SEC., AND HAS SEIZED IT AT REL. CLK.  2014 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2615  SEC., AND HAS SEIZED IT AT REL. CLK.  2615 FOR     1  SECONDS.

    1               2              SPS     DSC              2618                              0

    1               2              DSC     FMP              2619                              0

    2               2             GPH2     DSC              2622                              2

    2               2              DSC     FMP              2623                              0

    3              19              SPS     DSC              2628                              5

    3              19              DSC     FMP              2630                              0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2631  SEC., AND HAS SEIZED IT AT REL. CLK.  2631 FOR    60  SECONDS.

    8               6              FMP     DSC              2692                              0

    8               6              DSC     SPS              2695                              2

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2696  SEC., AND HAS SEIZED IT AT REL. CLK.  2696 FOR   180  SECONDS.

    7               4              SPS    GPH2              2880                              2

                      JOB # 64 COMPLETE AT RELATIVE CLOCK =  2881 SEC.
```

```
                                SUMMARY OF JOB #  65

FILE NO.      NUMBER OF           FROM     TO          TRANSFER COMPLETED AT          DURATION
        BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2882  SEC., AND HAS SEIZED IT AT REL. CLK.  2882 FOR     1  SECONDS.

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3483  SEC., AND HAS SEIZED IT AT REL. CLK.  3483 FOR     1  SECONDS.

    1             2              SPS     DSC              3485                           0

    1             2              DSC     FMP              3487                           0

    2             2             GPH2     DSC              3489                           2

    2             2              DSC     FMP              3491                           0

    3            19              SPS     DSC              3496                           5

    3            19              DSC     FMP              3498                           0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3499  SEC., AND HAS SEIZED IT AT REL. CLK.  3499 FOR    60  SECONDS.

    8             6              FMP     DSC              3560                           0

    8             6              DSC     SPS              3563                           2

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3564  SEC., AND HAS SEIZED IT AT REL. CLK.  3564 FOR   180  SECONDS.

    7             4              SPS    GPH2              3747                           2

                   JOB # 65 COMPLETE AT RELATIVE CLOCK =  3748 SEC.
```

```
                                    SUMMARY OF JOB #  66

FILE NO.      NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC).

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2965  SEC.. AND HAS SEIZED IT AT REL. CLK.  2965 FOR      1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3566  SEC.. AND HAS SEIZED IT AT REL. CLK.  3592 FOR      1  SECONDS.

    1             2               SPS      DSC              3595                      0

    1             2               DSC      FMP              3596                      0

    2             2              GPH1      DSC              3599                      3

    2             2               DSC      FMP              3601                      0

    3            19               SPS      DSC              3612                     11

    3            19               DSC      FMP              3614                      1

          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3615  SEC.. AND HAS SEIZED IT AT RFL. CLK.  3615 FOR     60  SECONDS.

    9           125               FMP      DSC              3681                      4

    9           125               FMP      DSC              3689                      7

    9           125               DSC      SPS              3751                     69

    9           125               DSC      SPS              3758                     67

                    JOB # 66 COMPLETE AT RELATIVE CLOCK =  3759 SEC.
```

```
                                  SUMMARY OF JOB #  67

FILE NO.       NUMBER OF         FROM      TO         TRANSFER COMPLETED AT          DURATION
        BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3077  SEC., AND HAS SEIZED IT AT REL. CLK.  3077 FOR     1  SECONDS.
   1              2             SPS      DSC              3080                          1
   1              2             DSC      FMP              3081                          0
   2              2            GPH2      SPS              3082                          1
       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3083  SEC., AND HAS SEIZED IT AT REL. CLK.  3083 FOR    80  SECONDS.
   2              2             SPS      DSC              3165                          0
   2              2             DSC      FMP              3166                          0
   3             19             SPS      DSC              3172                          5
   3             19             DSC      FMP              3174                          0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3175  SEC., AND HAS SEIZED IT AT REL. CLK.  3175 FOR    60  SECONDS.
   9            125             FMP      DSC              3240                          4
   9            125             FMP      DSC              3247                          5
   9            125             DSC      SPS              3309                         68
   9            125             DSC      SPS              3315                         67
                   JOB # 67 COMPLETE AT RELATIVE CLOCK =  3316 SEC.
```

```
                                          SUMMARY OF JOB #  68

FILE NO.        NUMBER OF            FROM      TO           TRANSFER COMPLETED AT            DURATION
           BLOCKS TRANSFERRED                                RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3280  SEC., AND HAS SEIZED IT AT REL. CLK.  3280 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3881  SEC., AND HAS SEIZED IT AT REL. CLK.  3881 FOR     1  SECONDS.
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4182  SEC., AND HAS SEIZED IT AT REL. CLK.  4182 FOR     1  SECONDS.
     1               2               SPS       DSC                4185                            0
     1               2               DSC       FMP                4186                            0
     2               2              GPH1       DSC                4189                            3
     2               2               DSC       FMP                4190                            0
     3              19               SPS       DSC                4195                            5
     3              19               DSC       FMP                4197                            0
          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4198  SEC., AND HAS SEIZED IT AT REL. CLK.  4515 FOR    60  SECONDS.
     8               6               FMP       DSC                4577                            0
     8               6               DSC       SPS                4580                            2
          REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4581  SEC., AND HAS SEIZED IT AT REL. CLK.  4631 FOR   180  SECONDS.
     7               4               SPS      GPH1                4814                            2
                         JOB # 68 COMPLETE AT RELATIVE CLOCK =  4815 SEC.
```

```
                                        SUMMARY OF JOB #  69

FILE NO.       NUMBER OF          FROM      TO           TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3338  SEC., AND HAS SEIZED IT AT REL. CLK.  3338 FOR     1  SECONDS.

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3939  SEC., AND HAS SEIZED IT AT REL. CLK.  3939 FOR     1  SECONDS.

   1             2               SPS       DSC              3942                           0

   1             2               DSC       FMP              3944                           0

   2             2               GPH2      DSC              3946                           2

   2             2               DSC       FMP              3947                           0

   3            19               SPS       DSC              3953                           5

   3            19               DSC       FMP              3955                           0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3956  SEC., AND HAS SEIZED IT AT REL. CLK.  4385 FOR    60  SECONDS.

   8             6               FMP       DSC              4447                           0

   8             6               DSC       SPS              4450                           1

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4451  SEC., AND HAS SEIZED IT AT REL. CLK.  4451 FOR   180  SECONDS.

   7             4               SPS      GPH2              4634                           2

                  JOB # 69 COMPLETE AT RELATIVE CLOCK =  4635 SEC.
```

```
                                   SUMMARY OF JOB #  70

FILE NO.        NUMBER OF          FROM       TO           TRANSFER COMPLETED AT            DURATION
         BLOCKS TRANSFERRED                               RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3530  SEC., AND HAS SEIZED IT AT REL. CLK.  3530 FOR     1  SECONDS.

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4131  SEC., AND HAS SEIZED IT AT REL. CLK.  4131 FOR     1  SECONDS.

    1              2               SPS       DSC                4134                          0

    1              2               DSC       FMP                4135                          0

    2              2              GPH1       DSC                4138                          3

    2              2               DSC       FMP                4140                          0

    3             19               SPS       DSC                4145                          5

    3             19               DSC       FMP                4147                          0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4148  SEC., AND HAS SEIZED IT AT REL. CLK.  4455 FOR    60  SECONDS.

    8             12               FMP       DSC                4517                          0

    8             12               DSC       SPS                4521                          3

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4522  SEC., AND HAS SEIZED IT AT REL. CLK.  4522 FOR   360  SECONDS.

    7              8               SPS      GPH1                4888                          4

                    JOB # 70 COMPLETE AT RELATIVE CLOCK =  4889 SEC.
```

```
                                   SUMMARY OF JOB #  71

FILE NO.        NUMBER OF           FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3729  SEC., AND HAS SEIZED IT AT REL. CLK.  3729 FOR     1  SECONDS.

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4330  SEC., AND HAS SEIZED IT AT REL. CLK.  4330 FOR     1  SECONDS.

    1              2                 SPS      DSC             4333                              0

    1              2                 DSC      FMP             4334                              0

    2              2                GPH2      DSC             4336                              2

    2              2                 DSC      FMP             4338                              0

    3             19                 SPS      DSC             4343                              5

    3             19                 DSC      FMP             4345                              0

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4346  SEC., AND HAS SEIZED IT AT REL. CLK.  5235 FOR    60  SECONDS.

    9            125                 FMP      DSC             5309                             12

    9            125                 FMP      DSC             5320                              9

    9            125                 DSC      SPS             5398                             87

    9            125                 DSC      SPS             5402                             80

                    JOB # 71 COMPLETE AT RELATIVE CLOCK =  5403 SEC.
```

```
                                    SUMMARY OF JOB #  72

FILE NO.       NUMBER OF           FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3799  SEC., AND HAS SEIZED IT AT REL. CLK.  3799 FOR     1  SECONDS.

    1              2                SPS      DSC                 3802                             0

    1              2                DSC      FMP                 3803                             0

    2              2               GPH1      SPS                 3804                             1

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3805  SEC., AND HAS SEIZED IT AT REL. CLK.  3805 FOR    80  SECONDS.

    2              2                SPS      DSC                 3886                             0

    2              2                DSC      FMP                 3888                             0

    3             19                SPS      DSC                 3893                             5

    3             19                DSC      FMP                 3895                             0

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3896  SEC., AND HAS SEIZED IT AT REL. CLK.  4315 FOR    60  SECONDS.

    8              6                FMP      DSC                 4377                             0

    8              6                DSC      SPS                 4380                             1

         REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4381  SEC., AND HAS SEIZED IT AT REL. CLK.  4381 FOR   180  SECONDS.

    7              4                SPS     GPH1                 4564                             2

                    JOB # 72 COMPLETE AT RELATIVE CLOCK =  4565 SEC.
```

```
                                    SUMMARY OF JOB #  73

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4152  SEC., AND HAS SEIZED IT AT REL. CLK.  4152 FOR      1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4753  SEC., AND HAS SEIZED IT AT REL. CLK.  4942 FOR      1  SECONDS.

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5243  SEC., AND HAS SEIZED IT AT REL. CLK.  5243 FOR      1  SECONDS.

    1               2             SPS       DSC                 5246                            0

    1               2             DSC       FMP                 5247                            0

    2               2            GPH1       DSC                 5254                            6

    2               2             DSC       FMP                 5255                            0

    3              19             SPS       DSC                 5262                            6

    3              19             DSC       FMP                 5264                            1

          REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5265  SEC., AND HAS SEIZED IT AT REL. CLK.  5995 FOR     60  SECONDS.

    8               6             FMP       DSC                 6057                            0

    8               6             DSC       SPS                 6127                            1

          REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6128  SEC., AND HAS SEIZED IT AT REL. CLK.  6279 FOR    180  SECONDS.

    7               6             SPS      GPH1                 6462                            2

                    JOB # 73 COMPLETE AT RELATIVE CLOCK =  6463 SEC.
```

```
                                   SUMMARY OF JOB #  74

FILE NO.        NUMBER OF          FROM     TO           TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                             RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5879  SEC.. AND HAS SEIZED IT AT REL. CLK.  5879 FOR     1  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6480  SEC.. AND HAS SEIZED IT AT REL. CLK.  6480 FOR     1  SECONDS.

    1             2                SPS      DSC              6482                            0

    1             2                DSC      FMP              6483                            0

    2             2                GPH1     DSC              6486                            2

    2             2                DSC      FMP              6487                            0

    3            19                SPS      DSC              6493                            6

    3            19                DSC      FMP              6495                            0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6496  SEC.. AND HAS SEIZED IT AT REL. CLK.  6855 FOR    60  SECONDS.

    8            12                FMP      DSC              6917                            0

    8            12                DSC      SPS              6922                            3

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6923  SEC.. AND HAS SEIZED IT AT REL. CLK.  6923 FOR   360  SECONDS.

    7             8                SPS      GPH1             7289                            4

                    JOB # 74 COMPLETE AT RELATIVE CLOCK =  7290 SEC.
```

```
                                        SUMMARY OF JOB #  75

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
         BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6148  SEC., AND HAS SEIZED IT AT REL. CLK.  6148 FOR     1  SECONDS.
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6749  SEC., AND HAS SEIZED IT AT REL. CLK.  6749 FOR     1  SECONDS.
       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7050  SEC., AND HAS SEIZED IT AT REL. CLK.  7268 FOR     1  SECONDS.
   1              2                 SPS      DSC              7271                            0
   1              2                 DSC      FMP              7273                            0
   2              2                GPH1      DSC              7275                            2
   2              2                 DSC      FMP              7276                            0
   3             19                 SPS      DSC              7282                            5
   3             19                 DSC      FMP              7283                            0
       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7284  SEC., AND HAS SEIZED IT AT REL. CLK.  7665 FOR    60  SECONDS.
```

```
                                   SUMMARY OF JOB #  76

FILE NO.       NUMBER OF         FROM     TO          TRANSFER COMPLETED AT            DURATION
         BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6215  SEC., AND HAS SEIZED IT AT REL. CLK.  6215 FOR     1  SECONDS.

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6816  SEC., AND HAS SEIZED IT AT REL. CLK.  7021 FOR     1  SECONDS.

    1                2            SPS     DSC              7024                             0

    1                2            DSC     FMP              7025                             0

    2                2           GPH1     DSC              7028                             2

    2                2            DSC     FMP              7029                             0

    3               19            SPS     DSC              7041                             5

    3               19            DSC     FMP              7042                             0

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7043  SEC., AND HAS SEIZED IT AT REL. CLK.  7585 FOR    60  SECONDS.

    9              125            FMP     DSC              7651                             4

    9              125            FMP     DSC              7658                             6
```

```
                                   SUMMARY OF JOB #  77

FILE NO.        NUMBER OF         FROM      TO          TRANSFER COMPLETED AT            DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6486  SEC., AND HAS SEIZED IT AT REL. CLK.  6486 FOR     1  SECONDS.
      1              2            SPS      DSC               6489                            0
      1              2            DSC      FMP               6490                            0
      2              2           GPH1      SPS               6491                            1
         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6492  SEC., AND HAS SEIZED IT AT RFL. CLK.  6492 FOR    80  SECONDS.
      2              2            SPS      DSC               6574                            0
      2              2            DSC      FMP               6575                            0
      3             19            SPS      DSC               6581                            5
      3             19            DSC      FMP               6583                            0
         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6584  SEC., AND HAS SEIZED IT AT RFL. CLK.  6925 FOR    60  SECONDS.
      8              6            FMP      DSC               6987                            0
      8              6            DSC      SPS               7023                            2
         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7024  SEC., AND HAS SEIZED IT AT REL. CLK.  7088 FOR   180  SECONDS.
      7              4            SPS     GPH1               7272                            2
                  JOB # 77 COMPLETE AT RELATIVE CLOCK =  7273 SEC.
```

```
                                   SUMMARY OF JOB #  78

FILE NO.        NUMBER OF           FROM      TO          TRANSFER COMPLETED AT             DURATION
           BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6522  SEC., AND HAS SEIZED IT AT REL. CLK.  6522 FOR      1  SECONDS.
           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7123  SEC., AND HAS SEIZED IT AT REL. CLK.  7269 FOR      1  SECONDS.
           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7570  SEC., AND HAS SEIZED IT AT REL. CLK.  7570 FOR      1  SECONDS.
    1              2                SPS       DSC              7573                        0
    1              2                DSC       FMP              7574                        0
    2              2                GPH1      DSC              7577                        2
    2              2                DSC       FMP              7578                        0
    3             19                SPS       DSC              7584                        5
    3             19                DSC       FMP              7586                        0
```

```
                                   SUMMARY OF JOB #  79

FILE NO.        NUMBER OF           FROM      TO          TRANSFER COMPLETED AT             DURATION
           BLOCKS TRANSFERRED                            RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6767  SEC., AND HAS SEIZED IT AT REL. CLK.  6767 FOR      1  SECONDS.
           REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7368  SEC., AND HAS SEIZED IT AT REL. CLK.  7368 FOR      1  SECONDS.
    1              2                SPS       DSC              7371                        0
    1              2                DSC       FMP              7372                        0
    2              2                GPH1      DSC              7375                        2
    2              2                DSC       FMP              7376                        0
    3             19                SPS       DSC              7382                        5
    3             19                DSC       FMP              7383                        0
```

```
                              SUMMARY OF JOB #  80

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7302  SEC.,  AND HAS SEIZED IT AT REL. CLK.  7302 FOR      1  SECONDS.



                              SUMMARY OF JOB #  81

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7595  SEC.,  AND HAS SEIZED IT AT REL. CLK.  7595 FOR      1  SECONDS.



                              SUMMARY OF JOB #  85

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
    1               4              SPS      DSC                  397                        1
    1               4              DSC      FMP                  398                        0
    2               4              GPH1     SPS                  401                        2
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  402  SEC.,  AND HAS SEIZED IT AT REL. CLK.   402 FOR    360  SECONDS.
    3              46              SPS      DSC                  777                       14
    3              46              DSC      FMP                  780                        2
    3              46              SPS      DSC                  792                       14
    3              46              DSC      FMP                  795                        1
        REQUESTED THE  FMP PROCESSOR AT REL. CLK.  796  SEC.,  AND HAS SEIZED IT AT REL. CLK.  1009 FOR     60  SECONDS.
    8               3              FMP      DSC                 1070                        0
    8               3              DSC      SPS                 1072                        1
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1073  SEC.,  AND HAS SEIZED IT AT REL. CLK.  1073 FOR    300  SECONDS.
    7               2              SPS      GPH1                1375                        1
                  JOB # 85 COMPLETE AT RELATIVE CLOCK =  1376 SEC.
```

```
                              SUMMARY OF JOB #  86

FILE NO.        NUMBER OF          FROM      TO         TRANSFER COMPLETED AT           DURATION
       BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

   1                  4             SPS     DSC               3684                        2

   1                  4             DSC     FMP               3686                        0

   2                  4             SPS     DSC               3687                        1

   2                  4             DSC     FMP               3689                        0

      REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3690  SEC., AND HAS SEIZED IT AT REL. CLK.  4275 FOR   10  SECONDS.

      REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4286  SEC., AND HAS SEIZED IT AT REL. CLK.  4575 FOR   60  SECONDS.

   9                160             FMP     DSC               4642                        5

   8                  3             FMP     DSC               4643                        0

   9                160             DSC     SPS               4689                       46

   8                  3             DSC     SPS               4690                        0

      REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4691  SEC., AND HAS SEIZED IT AT REL. CLK.  4811 FOR  300  SECONDS.

   7                  2             SPS    GPH2               5113                        1

              JOB # 86 COMPLETE AT RELATIVE CLOCK =  5114 SEC.
```

```
                                     SUMMARY OF JOB #  87

FILE NO.        NUMBER OF         FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1               4              SPS     DSC               4578                          1

    1               4              DSC     FMP               4579                          0

    2               4             GPH2     SPS               4581                          2

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4582  SEC., AND HAS SEIZED IT AT REL. CLK.  4582 FOR   360  SECONDS.

    3              92              SPS     DSC               4970                         26

    3              92              DSC     FMP               4974                          3

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4975  SEC., AND HAS SEIZED IT AT REL. CLK.  5915 FOR    60  SECONDS.

    8               3              FMP     DSC               5977                          0

    8               3              DSC     SPS               5979                          1

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5980  SEC., AND HAS SEIZED IT AT REL. CLK.  5980 FOR   300  SECONDS.

    7               2              SPS    GPH2               6282                          1

                      JOB # 87 COMPLETE AT RELATIVE CLOCK =  6283 SEC.
```

```
                                SUMMARY OF JOB #  88

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1              4               SPS      DSC              4890                          1

    1              4               DSC      FMP              4892                          0

    2              4               SPS      DSC              4892                          1

    2              4               DSC      FMP              4894                          0

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4895  SEC., AND HAS SEIZED IT AT REL. CLK.  5905 FOR    10  SECONDS.

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5916  SEC., AND HAS SEIZED IT AT REL. CLK.  6715 FOR    60  SECONDS.

    9            220               FMP      DSC              6784                          7

    8              3               FMP      DSC              6785                          0

    8              3               DSC      SPS              6787                          1

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6788  SEC., AND HAS SEIZED IT AT REL. CLK.  6788 FOR   300  SECONDS.

    7              2               SPS     GPH1              7091                          1

                      JOB # 88 COMPLETE AT RELATIVE CLOCK =  7092 SEC.




                                SUMMARY OF JOB #  89

FILE NO.       NUMBER OF          FROM      TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7357  SEC., AND HAS SEIZED IT AT REL. CLK.  7357 FOR     3  SECONDS.

    1              4               SPS      DSC              7363                          1

    1              4               DSC      FMP              7364                          0

    2              4               SPS      DSC              7366                          1

    2              4               DSC      FMP              7367                          0
```

```
                                      SUMMARY OF JOB #  92

FILE NO.         NUMBER OF          FROM      TO          TRANSFER COMPLETED AT              DURATION
         BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)

    1                4                SPS     DSC                   1                            1

    1                4                DSC     FMP                   2                            0

    2                4               GPH2     SPS                   4                            2

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK.    5  SEC.. AND HAS SEIZED IT AT REL. CLK.     5 FOR   360  SECONDS.

    3               92                SPS     DSC                 393                           27

    3               92                DSC     FMP                 398                            3

        REQUESTED THE  FMP PROCESSOR AT REL. CLK.  399  SEC.. AND HAS SEIZED IT AT REL. CLK.   399 FOR   600  SECONDS.

    8                8                FMP     DSC                1000                            0

    8                8                DSC     SPS                1004                            2

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1005  SEC.. AND HAS SEIZED IT AT REL. CLK.  1005 FOR   360  SECONDS.

        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1366  SEC.. AND HAS SEIZED IT AT REL. CLK.  1366 FOR   360  SECONDS.

    7                3                SPS    GPH2                1728                            1

                      JOB # 92 COMPLETE AT RELATIVE CLOCK =  1729 SEC.
```

```
                                          SUMMARY OF JOB #  93

FILE NO.       NUMBER OF           FROM       TO            TRANSFER COMPLETED AT               DURATION
          BLOCKS TRANSFERRED                               RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1                 4              SPS      DSC               1104                               1

    1                 4              DSC      FMP               1105                               0

    2                 4              SPS      DSC               1106                               1

    2                 4              DSC      FMP               1107                               0

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1108  SEC., AND HAS SEIZED IT AT REL. CLK.  1108 FOR    10  SECONDS.

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1119  SEC., AND HAS SEIZED IT AT REL. CLK.  1119 FOR   600  SECONDS.

    9               310              FMP      DSC               1731                              10

    8                 8              FMP      DSC               1732                               0

    8                 8              DSC      SPS               1736                               2

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1737  SEC., AND HAS SEIZED IT AT REL. CLK.  1737 FOR   360  SECONDS.

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2098  SEC., AND HAS SEIZED IT AT REL. CLK.  2098 FOR   360  SECONDS.

    7                 3              SPS     GPH2               2461                               1

                        JOB # 93 COMPLETE AT RELATIVE CLOCK =  2462 SEC.
```

```
                                   SUMMARY OF JOB #  94

FILE NO.       NUMBER OF           FROM     TO          TRANSFER COMPLETED AT          DURATION
          BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)
        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1171  SEC., AND HAS SEIZED IT AT REL. CLK.  1171 FOR     3  SECONDS.
   1              4                 SPS      DSC             1177                              1
   1              4                 DSC      FMP             1178                              0
   2              4                 SPS      DSC             1179                              1
   2              4                 DSC      FMP             1180                              0
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1181  SEC., AND HAS SEIZED IT AT REL. CLK.  1729 FOR   600  SECONDS.
   8              8                 FMP      DSC             2331                              0
   8              8                 DSC      SPS             2335                              3
        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2336  SEC., AND HAS SEIZED IT AT REL. CLK.  2336 FOR   360  SECONDS.
   9            160                 FMP      DSC             2337                              6
        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2697  SEC., AND HAS SEIZED IT AT REL. CLK.  2697 FOR   360  SECONDS.
   7              3                 SPS     GPH2             3060                              1
                 JOB # 94 COMPLETE AT RELATIVE CLOCK =  3061 SEC.
```

```
                                        SUMMARY OF JOB #  95

FILE NO.       NUMBER OF            FROM      TO          TRANSFER COMPLETED AT            DURATION
         BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...   OF TRANSFER (SEC)

    1               4                SPS      DSC                3228                          1
    1               4                DSC      FMP                3229                          0
    2               4               GPH1      SPS                3231                          2
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3232  SEC., AND HAS SEIZED IT AT REL. CLK.  3232 FOR   360  SECONDS.
    3              46                SPS      DSC                3615                         21
    3              46                DSC      FMP                3619                          2
    3              46                SPS      DSC                3630                         14
    3              46                DSC      FMP                3633                          1
        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3634  SEC., AND HAS SEIZED IT AT REL. CLK.  3675 FOR   600  SECONDS.
    8               8                FMP      DSC                4277                          0
    8               8                DSC      SPS                4280                          2
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4281  SEC., AND HAS SEIZED IT AT REL. CLK.  4281 FOR   360  SECONDS.
        REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4642  SEC., AND HAS SEIZED IT AT REL. CLK.  4642 FOR   360  SECONDS.
    7               3                SPS     GPH1                5005                          1
                JOB # 95 COMPLETE AT RELATIVE CLOCK =  5006 SEC.
```

```
                                     SUMMARY OF JOB #  96

FILE NO.       NUMBER OF         FROM      TO          TRANSFER COMPLETED AT           DURATION
        BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1                4            SPS      DSC               3755                         1

    1                4            DSC      FMP               3757                         0

    2                4            SPS      DSC               3758                         1

    2                4            DSC      FMP               3759                         0

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3760  SEC., AND HAS SEIZED IT AT REL. CLK.  4295 FOR    10  SECONDS.

         REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4306  SEC., AND HAS SEIZED IT AT REL. CLK.  4635 FOR   600  SECONDS.

    9              160            FMP      DSC               5242                         5

    8                8            FMP      DSC               5244                         0

    8                8            DSC      SPS               5249                         4

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 5250  SEC., AND HAS SEIZED IT AT REL. CLK.  5250 FOR   360  SECONDS.

    9              160            DSC      SPS               5343                        99

         REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 5611  SEC., AND HAS SEIZED IT AT REL. CLK.  5611 FOR   360  SECONDS.

    7                3            SPS     GPH2               5974                         1

                 JOB # 96 COMPLETE AT RELATIVE CLOCK =  5975 SEC.
```

```
                              SUMMARY OF JOB #  97

FILE NO.      NUMBER OF          FROM     TO          TRANSFER COMPLETED AT          DURATION
         BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1               4             SPS     DSC               3927                          1

    1               4             DSC     FMP               3929                          0

    2               4             SPS     DSC               3930                          1

    2               4             DSC     FMP               3931                          0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3932  SEC., AND HAS SEIZED IT AT RFL. CLK. 4375 FOR    10  SECONDS.

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4386  SEC., AND HAS SEIZED IT AT RFL. CLK.  5295 FOR   600  SECONDS.

    9             310             FMP     DSC               5911                         14

    8               8             FMP     DSC               5912                          0

    8               8             DSC     SPS               5918                          4

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5919  SEC., AND HAS SEIZED IT AT RFL. CLK.  5919 FOR   360  SECONDS.

       REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6280  SEC., AND HAS SEIZED IT AT RFL. CLK.  6280 FOR   360  SECONDS.

    7               3             SPS    GPH2               6643                          1

                  JOB # 97 COMPLETE AT RELATIVE CLOCK =  6644 SEC.
```

```
                                        SUMMARY OF JOB #  98

FILE NO.        NUMBER OF          FROM      TO          TRANSFER COMPLETED AT           DURATION
           BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

    1               4               SPS      DSC               4901                          1

    1               4               DSC      FMP               4902                          0

    2               4               GPH2     SPS               4905                          2

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4906  SEC., AND HAS SEIZED IT AT REL. CLK.  4906 FOR    360  SECONDS.

    3              46               SPS      DSC               5295                         28

    3              46               DSC      FMP               5303                          6

    3              46               SPS      DSC               5337                         40

    3              46               DSC      FMP               5343                          4

        REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5344  SEC., AND HAS SEIZED IT AT REL. CLK.  6055 FOR    600  SECONDS.

    8               8               FMP      DSC               6657                          0

    8               8               DSC      SPS               6660                          2

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6661  SEC., AND HAS SEIZED IT AT REL. CLK.  6661 FOR    360  SECONDS.

        REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7022  SEC., AND HAS SEIZED IT AT REL. CLK.  7024 FOR    360  SECONDS.

    7               3               SPS     GPH2               7387                          1

                    JOB # 98 COMPLETE AT RELATIVE CLOCK =  7388 SEC.
```

```
                                          SUMMARY OF JOB #  99

FILE NO.         NUMBER OF            FROM      TO          TRANSFER COMPLETED AT             DURATION
          BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME (SEC)...  OF TRANSFER (SEC)

   1                 4                 SPS      DSC              5946                          1

   1                 4                 DSC      FMP              5947                          0

   2                 4                 SPS      DSC              5948                          1

   2                 4                 DSC      FMP              5949                          0

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5950  SEC., AND HAS SEIZED IT AT REL. CLK.  6775 FOR   10  SECONDS.

       REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6786  SEC., AND HAS SEIZED IT AT REL. CLK.  6985 FOR  600  SECONDS.

   8                 8                 FMP      DSC              7587                          0

   8                 8                 DSC      SPS              7590                          2

       REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7591  SEC., AND HAS SEIZED IT AT REL. CLK.  7591 FOR  360  SECONDS.
```

SUMMARY OF FILE TRANSFER REQUESTS

```
JOB NO.        FILE NO.     NUMBER OF        FROM      TO        TRANSFER COMPLETED AT            DURATION
                        BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

   92              1            4             SPS      DSC                   1                      1

   92              1            4             DSC      FMP                   2                      0

   92              2            4            GPH2      SPS                   4                      2

          JOB #  92 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.    5 SEC., AND HAS SEIZED IT AT REL. CLK.    5 FOR  360 SECONDS.

          JOB #   1 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  157 SEC., AND HAS SEIZED IT AT REL. CLK.  157 FOR    1 SECONDS.

          JOB #   2 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  251 SEC., AND HAS SEIZED IT AT REL. CLK.  251 FOR    1 SECONDS.

          JOB #   3 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  354 SEC., AND HAS SEIZED IT AT REL. CLK.  354 FOR    1 SECONDS.

    3              1            1             SPS      DSC                 356                      0

    3              1            1             DSC      FMP                 357                      0

    3              2            1            GPH1      DSC                 359                      1

    3              2            1             DSC      FMP                 360                      0

          JOB #   3 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  361 SEC., AND HAS SEIZED IT AT REL. CLK.  361 FOR   10 SECONDS.

    3              8            2             FMP      DSC                 372                      0

    3              8            2             DSC      SPS                 374                      0

          JOB #   3 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  375 SEC., AND HAS SEIZED IT AT REL. CLK.  375 FOR   60 SECONDS.

          JOB #   4 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  390 SEC., AND HAS SEIZED IT AT REL. CLK.  390 FOR    1 SECONDS.

   92              3           92             SPS      DSC                 393                     27

   85              1            4             SPS      DSC                 397                      1

   92              3           92             DSC      FMP                 398                      3
```

```
                                      FILE TRANSFER SUMMARY - PAGE  2

JOB NO.        FILE NO.      NUMBER OF          FROM        TO         TRANSFER COMPLETED AT            DURATION
                          BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...    OF TRANSFER (SEC.)

  85              1              4              DSC        FMP                   398                        0
     JOB #  92 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  399 SEC., AND HAS SEIZED IT AT REL. CLK.  399 FOR  600 SECONDS.
  85              2              4              GPH1       SPS                   401                        2
     JOB #  85 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  402 SEC., AND HAS SEIZED IT AT REL. CLK.  402 FOR  360 SECONDS.
                          JOB #  3 COMPLETE AT RELATIVE CLOCK =   436 SEC.
     JOB #   5 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  745 SEC., AND HAS SEIZED IT AT REL. CLK.  745 FOR    1 SECONDS.
     JOB #   1 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  758 SEC., AND HAS SEIZED IT AT REL. CLK.  758 FOR    1 SECONDS.
   1              1              1              SPS        DSC                   761                        0
   1              1              1              DSC        FMP                   762                        0
   1              2              1              GPH2       DSC                   764                        1
   1              2              1              DSC        FMP                   765                        0
     JOB #   6 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  775 SEC., AND HAS SEIZED IT AT REL. CLK.  775 FOR    1 SECONDS.
  85              3             46              SPS        DSC                   777                       14
  85              3             46              DSC        FMP                   780                        2
  85              3             46              SPS        DSC                   792                       14
  85              3             46              DSC        FMP                   795                        1
     JOB #  61 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  813 SEC., AND HAS SEIZED IT AT REL. CLK.  813 FOR    1 SECONDS.
     JOB #   2 REQUESTED THE SPS2 PROCESSOR AT REL. CLK.  852 SEC., AND HAS SEIZED IT AT REL. CLK.  852 FOR    1 SECONDS.
     JOB #   7 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  940 SEC., AND HAS SEIZED IT AT REL. CLK.  940 FOR    1 SECONDS.
   7              1              1              SPS        DSC                   942                        0
   7              1              1              DSC        FMP                   943                        0
   7              2              1              GPH2       DSC                   945                        1
```

```
                                    FILE TRANSFER SUMMARY - PAGE  3

JOB NO.          FILE NO.       NUMBER OF           FROM        TO         TRANSFER COMPLETED AT            DURATION
                             BLOCKS TRANSFERRED                           RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

   7                  2              1               DSC         FMP                  946                         0

       JOB #   4 REQUESTED THE SPS1 PROCESSOR AT REL. CLK.  991 SEC.. AND HAS SEIZED IT AT REL. CLK.  991 FOR    1 SECONDS.

   4                  1              1               SPS         DSC                  993                         0

   4                  1              1               DSC         FMP                  994                         0

   4                  2              1               GPH1        DSC                  996                         1

   4                  2              1               DSC         FMP                  997                         0

       JOB #   1 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  766 SEC.. AND HAS SEIZED IT AT REL. CLK.  999 FOR   10 SECONDS.

  92                  8              8               FMP         DSC                 1000                         0

  92                  8              8               DSC         SPS                 1004                         2

       JOB #  92 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1005 SEC.. AND HAS SEIZED IT AT REL. CLK. 1005 FOR  360 SECONDS.

       JOB #  85 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  796 SEC.. AND HAS SEIZED IT AT REL. CLK. 1009 FOR   60 SECONDS.

   1                  8              2               FMP         DSC                 1010                         0

   1                  8              2               DSC         SPS                 1012                         0

       JOB #   1 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1013 SEC.. AND HAS SEIZED IT AT REL. CLK. 1013 FOR   60 SECONDS.

       JOB #   8 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1032 SEC.. AND HAS SEIZED IT AT REL. CLK. 1032 FOR    1 SECONDS.

       JOB #   7 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  947 SEC.. AND HAS SEIZED IT AT REL. CLK. 1069 FOR   10 SECONDS.

  85                  8              3               FMP         DSC                 1070                         0

  85                  8              3               DSC         SPS                 1072                         1

       JOB #  85 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1073 SEC.. AND HAS SEIZED IT AT REL. CLK. 1073 FOR  300 SECONDS.

                         JOB #  1 COMPLETE AT RELATIVE CLOCK =  1074 SEC.

       JOB #   4 REQUESTED THE  FMP PROCESSOR AT REL. CLK.  998 SEC.. AND HAS SEIZED IT AT REL. CLK. 1079 FOR   10 SECONDS.

   7                  8              2               FMP         DSC                 1080                         0

   7                  8              2               DSC         SPS                 1082                         0

       JOB #   7 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1083 SEC.. AND HAS SEIZED IT AT REL. CLK. 1083 FOR   60 SECONDS.
```

```
                              FILE TRANSFER SUMMARY - PAGE  4

JOB NO.        FILE NO.      NUMBER OF          FROM      TO       TRANSFER COMPLETED AT        DURATION
                          BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

          JOB #   9 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1087 SEC., AND HAS SEIZED IT AT REL. CLK. 1087 FOR    1 SECONDS.
   4               9            90           FMP       DSC               1093                     3
  93               1             4           SPS       DSC               1104                     1
  93               1             4           DSC       FMP               1105                     0
  93               2             4           SPS       DSC               1106                     1
          JOB #  10 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1106 SEC., AND HAS SEIZED IT AT REL. CLK. 1106 FOR   <1 SECONDS.
  93               2             4           DSC       FMP               1107                     0
          JOB #  93 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1108 SEC., AND HAS SEIZED IT AT REL. CLK. 1108 FOR   10 SECONDS.
          JOB #  93 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1119 SEC., AND HAS SEIZED IT AT REL. CLK. 1119 FOR  600 SECONDS.
                         JOB #  7 COMPLETE AT RELATIVE CLOCK =  1144 SEC.
          JOB #   2 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1153 SEC., AND HAS SEIZED IT AT REL. CLK. 1153 FOR    1 SECONDS.
   2               1             1           SPS       DSC               1155                     0
   2               1             1           DSC       FMP               1156                     0
   2               2             1           GPH1      DSC               1157                     1
   2               2             1           DSC       FMP               1159                     0
   4               9            90           DSC       SPS               1170                    26
          JOB #   4 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1171 SEC., AND HAS SEIZED IT AT REL. CLK. 1171 FOR   60 SECONDS.
          JOB #  94 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1171 SEC., AND HAS SEIZED IT AT REL. CLK. 1171 FOR    3 SECONDS.
  94               1             4           SPS       DSC               1177                     1
  94               1             4           DSC       FMP               1178                     0
  94               2             4           SPS       DSC               1179                     1
  94               2             4           DSC       FMP               1180                     0
                         JOB #  4 COMPLETE AT RELATIVE CLOCK =  1232 SEC.
```

```
                              FILE TRANSFER SUMMARY - PAGE  5

JOB NO.         FILE NO.       NUMBER OF          FROM      TO        TRANSFER COMPLETED AT          DURATION
                           BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

          JOB #  11 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1297 SEC., AND HAS SEIZED IT AT REL. CLK. 1297 FOR    1 SECONDS.
   11              1             1               SPS       DSC                1299                      0
   11              1             1               DSC       FMP                1300                      0
   11              2             1               GPH2      DSC                1301                      1
   11              2             1               DSC       FMP                1303                      0
          JOB #   5 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1346 SEC., AND HAS SEIZED IT AT REL. CLK. 1346 FOR    1 SECONDS.
          JOB #  62 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1349 SEC., AND HAS SEIZED IT AT REL. CLK. 1349 FOR    1 SECONDS.
    5              1             1               SPS       DSC                1349                      0
    5              1             1               DSC       FMP                1350                      0
    5              2             1               GPH2      DSC                1351                      1
   62              1             2               SPS       DSC                1351                      0
    5              2             1               DSC       FMP                1352                      0
   62              1             2               DSC       FMP                1353                      0
   62              2             2               GPH1      SPS                1354                      1
          JOB #  62 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1355 SEC., AND HAS SEIZED IT AT REL. CLK. 1355 FOR   80 SECONDS.
          JOB #  63 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1355 SEC., AND HAS SEIZED IT AT REL. CLK. 1355 FOR    1 SECONDS.
          JOB #  92 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1366 SEC., AND HAS SEIZED IT AT REL. CLK. 1366 FOR  360 SECONDS.
   85              7             2               SPS       GPH1               1375                      1
          JOB #   6 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1376 SEC., AND HAS SEIZED IT AT REL. CLK. 1376 FOR    1 SECONDS.
                            JOB # 85 COMPLETE AT RELATIVE CLOCK =  1376 SEC.
          JOB #  61 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1414 SEC., AND HAS SEIZED IT AT REL. CLK. 1414 FOR    1 SECONDS.
   61              1             2               SPS       DSC                1417                      0
   61              1             2               DSC       FMP                1418                      0
```

```
                              FILE TRANSFER SUMMARY - PAGE  6

JOB NO.        FILE NO.      NUMBER OF         FROM      TO       TRANSFER COMPLETED AT         DURATION
                           BLOCKS TRANSFERRED                    RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  61               2            2              GPH2      DSC              1421                      2
  61               2            2               DSC      FMP              1422                      0
  61               3           19               SPS      DSC              1427                      5
  61               3           19               DSC      FMP              1429                      0
  62               2            2               SPS      DSC              1436                      0
  62               2            2               DSC      FMP              1438                      0
  62               3           19               SPS      DSC              1443                      5
  62               3           19               DSC      FMP              1445                      0
     JOB #  12 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1487 SEC., AND HAS SEIZED IT AT REL. CLK. 1487 FOR    1 SECONDS.
     JOB #   8 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1633 SEC., AND HAS SEIZED IT AT REL. CLK. 1633 FOR    1 SECONDS.
   8               1            1               SPS      DSC              1635                      0
   8               1            1               DSC      FMP              1636                      0
   8               2            1              GPH2      DSC              1637                      1
   8               2            1               DSC      FMP              1639                      0
     JOB #   6 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1677 SEC., AND HAS SEIZED IT AT REL. CLK. 1677 FOR    1 SECONDS.
   6               1            1               SPS      DSC              1680                      0
   6               1            1               DSC      FMP              1681                      0
   6               2            1              GPH1      DSC              1682                      1
   6               2            1               DSC      FMP              1683                      0
     JOB #  13 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1686 SEC., AND HAS SEIZED IT AT REL. CLK. 1686 FOR    1 SECONDS.
     JOB #   9 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1688 SEC., AND HAS SEIZED IT AT REL. CLK. 1688 FOR    1 SECONDS.
   9               1            1               SPS      DSC              1690                      0
   9               1            1               DSC      FMP              1692                      0
   9               2            1              GPH2      DSC              1693                      1
```

```
                         FILE TRANSFER SUMMARY - PAGE  7

JOB NO.        FILE NO.      NUMBER OF         FROM       TO       TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

   9              2              1              DSC       FMP               1694                     0
       JOB #  10 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1707 SEC., AND HAS SEIZED IT AT REL. CLK. 1707 FOR    1 SECONDS.
       JOB #   2 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1160 SEC., AND HAS SEIZED IT AT REL. CLK. 1719 FOR   10 SECONDS.
  92              7              3              SPS       GPH2              1728                     1
       JOB #  94 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1181 SEC., AND HAS SEIZED IT AT REL. CLK. 1729 FOR  600 SECONDS.
                            JOB # 92 COMPLETE AT RELATIVE CLOCK =  1729 SEC.
   2              8              2              FMP       DSC               1730                     0
  93              9            310              FMP       DSC               1731                    10
   2              8              2              DSC       SPS               1732                     0
  93              8              8              FMP       DSC               1732                     0
       JOB #   2 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1733 SEC., AND HAS SEIZED IT AT REL. CLK. 1733 FOR   60 SECONDS.
  93              8              8              DSC       SPS               1736                     2
       JOB #  93 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1737 SEC., AND HAS SEIZED IT AT REL. CLK. 1737 FOR  360 SECONDS.
                            JOB #  2 COMPLETE AT RELATIVE CLOCK =  1794 SEC.
       JOB #  14 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1801 SEC., AND HAS SEIZED IT AT REL. CLK. 1801 FOR    1 SECONDS.
       JOB #  15 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 1816 SEC., AND HAS SEIZED IT AT REL. CLK. 1816 FOR    1 SECONDS.
  15              1              1              SPS       DSC               1819                     0
  15              1              1              DSC       FMP               1820                     0
  15              2              1              GPH2      DSC               1821                     1
  15              2              1              DSC       FMP               1823                     0
       JOB #  63 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 1956 SEC., AND HAS SEIZED IT AT REL. CLK. 1956 FOR    1 SECONDS.
       JOB #  10 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2008 SEC., AND HAS SEIZED IT AT REL. CLK. 2008 FOR    1 SECONDS.
  10              1              1              SPS       DSC               2011                     0
  10              1              1              DSC       FMP               2012                     0
```

```
                              FILE TRANSFER SUMMARY - PAGE  8

 JOB NO.        FILE NO.      NUMBER OF          FROM       TO       TRANSFER COMPLETED AT           DURATION
                           BLOCKS TRANSFERRED                       RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

   10              2               1             GPH1       DSC              2013                      1
        JOB #  64 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2014 SEC., AND HAS SEIZED IT AT REL. CLK. 2014 FOR    1 SECONDS.
   10              2               1             DSC        FMP              2015                      0
        JOB #  16 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2039 SEC., AND HAS SEIZED IT AT REL. CLK. 2039 FOR    1 SECONDS.
        JOB #  17 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2051 SEC., AND HAS SEIZED IT AT REL. CLK. 2051 FOR    1 SECONDS.
        JOB #  12 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2088 SEC., AND HAS SEIZED IT AT REL. CLK. 2088 FOR    1 SECONDS.
   12              1               1             SPS        DSC              2090                      0
   12              1               1             DSC        FMP              2091                      0
   12              2               1             GPH2       DSC              2093                      1
   12              2               1             DSC        FMP              2094                      0
        JOB #  93 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2098 SEC., AND HAS SEIZED IT AT REL. CLK. 2098 FOR  360 SECONDS.
        JOB #  18 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2148 SEC., AND HAS SEIZED IT AT REL. CLK. 2148 FOR    1 SECONDS.
        JOB #  19 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2223 SEC., AND HAS SEIZED IT AT REL. CLK. 2223 FOR    1 SECONDS.
   19              1               1             SPS        DSC              2225                      0
   19              1               1             DSC        FMP              2226                      0
   19              2               1             GPH2       DSC              2228                      1
   19              2               1             DSC        FMP              2229                      0
        JOB #  63 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2257 SEC., AND HAS SEIZED IT AT REL. CLK. 2257 FOR    1 SECONDS.
   63              1               2             SPS        DSC              2260                      0
   63              1               2             DSC        FMP              2261                      0
```

```
                              FILE TRANSFER SUMMARY - PAGE  9

JOB NO.          FILE NO.        NUMBER OF          FROM       TO          TRANSFER COMPLETED AT          DURATION
                              BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

          JOB #  20 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2262 SEC., AND HAS SEIZED IT AT REL. CLK. 2262 FOR    1 SECONDS.
   63              2               2            GPH2       DSC                2264                        3
   63              2               2            DSC        FMP                2265                        0
   63              3              19            SPS        DSC                2271                        5
   63              3              19            DSC        FMP                2272                        0
          JOB #  13 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2287 SEC., AND HAS SEIZED IT AT REL. CLK. 2287 FOR    1 SECONDS.
   13              1               1            SPS        DSC                2289                        0
   13              1               1            DSC        FMP                2291                        0
   13              2               1            GPH1       DSC                2292                        1
   13              2               1            DSC        FMP                2293                        0
          JOB #  11 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1304 SEC., AND HAS SEIZED IT AT REL. CLK. 2329 FOR   10 SECONDS.
   94              8               8            FMP        DSC                2331                        0
   94              8               8            DSC        SPS                2335                        3
          JOB #  94 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2336 SEC., AND HAS SEIZED IT AT REL. CLK. 2336 FOR  360 SECONDS.
   94              9             160            FMP        DSC                2337                        6
          JOB #   5 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1353 SEC., AND HAS SEIZED IT AT REL. CLK. 2339 FOR   10 SECONDS.
   11              9              90            FMP        DSC                2343                        3
          JOB #  61 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1430 SEC., AND HAS SEIZED IT AT REL. CLK. 2349 FOR   60 SECONDS.
    5              8               2            FMP        DSC                2350                        0
    5              8               2            DSC        SPS                2352                        0
   11              9              90            DSC        SPS                2372                       27
          JOB #   5 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2353 SEC., AND HAS SEIZED IT AT REL. CLK. 2372 FOR   60 SECONDS.
          JOB #  14 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2402 SEC., AND HAS SEIZED IT AT REL. CLK. 2402 FOR    1 SECONDS.
          JOB #  62 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1446 SEC., AND HAS SEIZED IT AT REL. CLK. 2409 FOR   60 SECONDS.
   61              9             125            FMP        DSC                2415                        4
   61              9             125            FMP        DSC                2421                        5
```

```
                              FILE TRANSFER SUMMARY - PAGE 10

JOB NO.         FILE NO.       NUMBER OF        FROM      TO       TRANSFER COMPLETED AT          DURATION
                           BLOCKS TRANSFERRED                     RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

      JOB #  11 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2373 SEC.. AND HAS SEIZED IT AT REL. CLK. 2432 FOR   60 SECONDS.
                             JOB #  5 COMPLETE AT RELATIVE CLOCK =  2433 SEC.
  93                7              3             SPS      GPH2                 2461                      1
                             JOB # 93 COMPLETE AT RELATIVE CLOCK =  2462 SEC.
      JOB #   8 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1640 SEC.. AND HAS SEIZED IT AT REL. CLK. 2469 FOR   10 SECONDS.
  62                8              6             FMP      DSC                  2471                      0
  62                8              6             DSC      SPS                  2475                      2
      JOB #  62 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2476 SEC.. AND HAS SEIZED IT AT REL. CLK. 2476 FOR  180 SECONDS.
      JOB #   6 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1684 SEC.. AND HAS SEIZED IT AT REL. CLK. 2479 FOR   10 SECONDS.
  61                9            125             DSC      SPS                  2485                     69
   8                9             90             FMP      DSC                  2486                      5
      JOB #   9 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1695 SEC.. AND HAS SEIZED IT AT REL. CLK. 2489 FOR   10 SECONDS.
   6                8              2             FMP      DSC                  2490                      0
  61                9            125             DSC      SPS                  2490                     67
                             JOB # 61 COMPLETE AT RELATIVE CLOCK =  2491 SEC.
                             JOB # 11 COMPLETE AT RELATIVE CLOCK =  2493 SEC.
      JOB #  15 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 1824 SEC.. AND HAS SEIZED IT AT REL. CLK. 2499 FOR   10 SECONDS.
   9                8              2             FMP      DSC                  2500                      0
      JOB #  10 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2016 SEC.. AND HAS SEIZED IT AT REL. CLK. 2509 FOR   10 SECONDS.
  15                8              2             FMP      DSC                  2510                      0
  15                8              2             DSC      SPS                  2512                      0
      JOB #  15 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2513 SEC.. AND HAS SEIZED IT AT REL. CLK. 2513 FOR   60 SECONDS.
   8                9             90             DSC      SPS                  2518                     26
      JOB #  12 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2095 SEC.. AND HAS SEIZED IT AT REL. CLK. 2519 FOR   10 SECONDS.
   6                8              2             DSC      SPS                  2519                      0
   9                8              2             DSC      SPS                  2520                      0
```

```
                                   FILE TRANSFER SUMMARY - PAGE 11

JOB NO.         FILE NO.       NUMBER OF          FROM       TO         TRANSFER COMPLETED AT          DURATION
                           BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

    JOB #   8 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2519 SEC.. AND HAS SEIZED IT AT REL. CLK. 2520 FOR   60 SECONDS.
 10              8              2            FMP        DSC               2520                       0
    JOB #  19 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2230 SEC.. AND HAS SEIZED IT AT REL. CLK. 2529 FOR   10 SECONDS.
 12              9             90            FMP        DSC               2533                       3
    JOB #  63 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2273 SEC.. AND HAS SEIZED IT AT RFL. CLK. 2539 FOR   60 SECONDS.
 19              9             90            FMP        DSC               2543                       3
 19              9             90            DSC        SPS               2571                      26
    JOB #  19 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2572 SEC.. AND HAS SEIZED IT AT REL. CLK. 2572 FOR   60 SECONDS.
                   JOB # 15 COMPLETE AT RELATIVE CLOCK =  2574 SEC.
    JOB #   6 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2520 SEC.. AND HAS SEIZED IT AT REL. CLK. 2580 FOR   60 SECONDS.
                   JOB #  8 COMPLETE AT RELATIVE CLOCK =  2581 SEC.
    JOB #  13 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2294 SEC.. AND HAS SEIZED IT AT REL. CLK. 2599 FOR   10 SECONDS.
 63              8             12            FMP        DSC               2601                       0
 13              8              2            FMP        DSC               2610                       0
    JOB #  64 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2615 SEC.. AND HAS SEIZED IT AT REL. CLK. 2615 FOR    1 SECONDS.
 64              1              2            SPS        DSC               2618                       0
 64              1              2            DSC        FMP               2619                       0
 64              2              2            GPH2       DSC               2622                       2
 64              2              2            DSC        FMP               2623                       0
 64              3             19            SPS        DSC               2628                       5
 64              3             19            DSC        FMP               2630                       0
    JOB #  64 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2631 SEC.. AND HAS SEIZED IT AT REL. CLK. 2631 FOR   60 SECONDS.
                   JOB # 19 COMPLETE AT RELATIVE CLOCK =  2633 SEC.
    JOB #  16 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2640 SEC.. AND HAS SEIZED IT AT REL. CLK. 2640 FOR    1 SECONDS.
    JOB #   9 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2521 SEC.. AND HAS SEIZED IT AT REL. CLK. 2640 FOR   60 SECONDS.
                   JOB #  6 COMPLETE AT RELATIVE CLOCK =  2641 SEC.
```

```
                              FILE TRANSFER SUMMARY - PAGE 12

JOB NO.       FILE NO.       NUMBER OF        FROM      TO        TRANSFER COMPLETED AT       DURATION
                         BLOCKS TRANSFERRED                     RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

  16             1              1              SPS      DSC              2642                     0

  16             1              1              DSC      FMP              2644                     0

  16             2              1             GPH1      DSC              2645                     1

      JOB #  21 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2646 SEC., AND HAS SEIZED IT AT REL. CLK. 2646 FOR    1 SECONDS.

  16             2              1              DSC      FMP              2646                     0

  21             1              1              SPS      DSC              2648                     0

  21             1              1              DSC      FMP              2649                     0

  21             2              1             GPH1      DSC              2650                     1

  21             2              1              DSC      FMP              2652                     0

      JOB #  17 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2652 SEC., AND HAS SEIZED IT AT REL. CLK. 2652 FOR    1 SECONDS.

  17             1              1              SPS      DSC              2654                     0

  17             1              1              DSC      FMP              2656                     0

  10             8              2              DSC      SPS              2657                     1

  17             2              1             GPH2      DSC              2658                     2

  17             2              1              DSC      FMP              2659                     0

  62             7              4              SPS     GPH1              2660                     2

                           JOB # 62 COMPLETE AT RELATIVE CLOCK =  2661 SEC.

  13             8              2              DSC      SPS              2661                     1

      JOB #  10 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2658 SEC., AND HAS SEIZED IT AT REL. CLK. 2661 FOR   60 SECONDS.

  63             8             12              DSC      SPS              2664                     8

      JOB #  13 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2662 SEC., AND HAS SEIZED IT AT REL. CLK. 2664 FOR   60 SECONDS.

  12             9             90              DSC      SPS              2687                    31

      JOB #  12 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2688 SEC., AND HAS SEIZED IT AT REL. CLK. 2688 FOR   60 SECONDS.

      JOB #  16 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2647 SEC., AND HAS SEIZED IT AT REL. CLK. 2691 FOR   10 SECONDS.

  64             8              6              FMP      DSC              2692                     0
```

```
                              FILE TRANSFER SUMMARY - PAGE 13

JOB NO.        FILE NO.      NUMBER OF          FROM       TO        TRANSFER COMPLETED AT           DURATION
                         BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  64              8              6              DSC        SPS                2695                       2
        JOB #  64 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2696 SEC.. AND HAS SEIZED IT AT REL. CLK. 2696 FOR  180 SECONDS.
        JOB #  94 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2697 SEC.. AND HAS SEIZED IT AT REL. CLK. 2697 FOR  360 SECONDS.
        JOB #  21 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2653 SEC.. AND HAS SEIZED IT AT REL. CLK. 2701 FOR   10 SECONDS.
                          JOB #  9 COMPLETE AT RELATIVE CLOCK =  2701 SEC.
        JOB #  14 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2703 SEC.. AND HAS SEIZED IT AT REL. CLK. 2703 FOR    1 SECONDS.
  16              9             90              FMP        DSC                2705                       3
  16              1              1              SPS        DSC                2706                       0
  14              1              1              DSC        FMP                2708                       0
  14              2              1              GPH2       DSC                2710                       2
        JOB #  17 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2660 SEC.. AND HAS SEIZED IT AT REL. CLK. 2711 FOR   10 SECONDS.
  14              2              1              DSC        FMP                2711                       0
  21              8              2              FMP        DSC                2712                       0
  21              8              2              DSC        SPS                2715                       1
        JOB #  21 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2716 SEC.. AND HAS SEIZED IT AT REL. CLK. 2716 FOR   60 SECONDS.
        JOB #  14 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2712 SEC.. AND HAS SEIZED IT AT REL. CLK. 2721 FOR   10 SECONDS.
                          JOB # 10 COMPLETE AT RELATIVE CLOCK =  2722 SEC.
  17              8              2              FMP        DSC                2722                       0
        JOB #  63 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 2665 SEC.. AND HAS SEIZED IT AT REL. CLK. 2724 FOR  360 SECONDS.
  17              8              2              DSC        SPS                2725                       1
                          JOB # 13 COMPLETE AT RELATIVE CLOCK =  2725 SEC.
        JOB #  17 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2726 SEC.. AND HAS SEIZED IT AT REL. CLK. 2726 FOR   60 SECONDS.
  14              8              2              FMP        DSC                2732                       0
  14              8              2              DSC        SPS                2734                       0
  16              9             90              DSC        SPS                2736                      30
        JOB #  14 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2735 SEC.. AND HAS SEIZED IT AT REL. CLK. 2736 FOR   60 SECONDS.
```

```
                              FILE TRANSFER SUMMARY - PAGE 14

JOB NO.         FILE NO.        NUMBER OF          FROM       TO        TRANSFER COMPLETED AT         DURATION
                            BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

                         JOB # 12 COMPLETE AT RELATIVE CLOCK =  2749 SEC.

        JOB #  16 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2737 SEC., AND HAS SEIZED IT AT REL. CLK. 2776 FOR   60 SECONDS.

                         JOB # 21 COMPLETE AT RELATIVE CLOCK =  2777 SEC.

        JOB #  18 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2749 SEC., AND HAS SEIZED IT AT REL. CLK. 2786 FOR    1 SECONDS.

                         JOB # 17 COMPLETE AT RELATIVE CLOCK =  2787 SEC.

                         JOB # 14 COMPLETE AT RELATIVE CLOCK =  2797 SEC.

        JOB #  22 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2831 SEC., AND HAS SEIZED IT AT REL. CLK. 2831 FOR    1 SECONDS.

   22               1             1            SPS       DSC                 2833                       0

   22               1             1            DSC       FMP                 2834                       0

   22               2             1           GPH1       DSC                 2835                       1

                         JOB # 16 COMPLETE AT RELATIVE CLOCK =  2837 SEC.

   22               2             1            DSC       FMP                 2837                       0

        JOB #  22 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2838 SEC., AND HAS SEIZED IT AT REL. CLK. 2838 FOR   10 SECONDS.

   22               8             2            FMP       DSC                 2849                       0

   22               8             2            DSC       SPS                 2850                       0

        JOB #  22 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2851 SEC., AND HAS SEIZED IT AT REL. CLK. 2851 FOR   60 SECONDS.

        JOB #  20 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2863 SEC., AND HAS SEIZED IT AT REL. CLK. 2863 FOR    1 SECONDS.

   20               1             1            SPS       DSC                 2866                       0

   20               1             1            DSC       FMP                 2867                       0

   20               2             1           GPH1       DSC                 2868                       1

   20               2             1            DSC       FMP                 2870                       0

        JOB #  20 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2871 SEC., AND HAS SEIZED IT AT REL. CLK. 2871 FOR   10 SECONDS.

   64               7             4            SPS      GPH2                 2880                       2

                         JOB # 64 COMPLETE AT RELATIVE CLOCK =  2881 SEC.
```

```
                                   FILE TRANSFER SUMMARY - PAGE 15

JOB NO.          FILE NO.        NUMBER OF          FROM        TO         TRANSFER COMPLETED AT         DURATION
                           BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

       JOB #  65 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2882 SEC.. AND HAS SEIZED IT AT REL. CLK. 2882 FOR    1 SECONDS.
  20                9             90             FMP         DSC                  2885                        3
  20                9             90             DSC         SPS                  2912                       26
                        JOB # 22 COMPLETE AT RELATIVE CLOCK =  2912 SEC.
       JOB #  20 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2913 SEC.. AND HAS SEIZED IT AT REL. CLK. 2913 FOR   60 SECONDS.
       JOB #  23 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2958 SEC.. AND HAS SEIZED IT AT REL. CLK. 2958 FOR    1 SECONDS.
  23                1              1             SPS         DSC                  2961                        0
  23                1              1             DSC         FMP                  2962                        0
  23                2              1             GPH1        DSC                  2963                        1
  23                2              1             DSC         FMP                  2964                        0
       JOB #  23 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 2965 SEC.. AND HAS SEIZED IT AT REL. CLK. 2965 FOR   10 SECONDS.
       JOB #  66 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2965 SEC.. AND HAS SEIZED IT AT REL. CLK. 2965 FOR    1 SECONDS.
                        JOB # 20 COMPLETE AT RELATIVE CLOCK =  2974 SEC.
  23                8              2             FMP         DSC                  2977                        0
  23                8              2             DSC         SPS                  2978                        0
       JOB #  23 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 2979 SEC.. AND HAS SEIZED IT AT REL. CLK. 2979 FOR   60 SECONDS.
       JOB #  24 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3031 SEC.. AND HAS SEIZED IT AT REL. CLK. 3031 FOR    1 SECONDS.
  24                1              1             SPS         DSC                  3034                        0
  24                1              1             DSC         FMP                  3035                        0
  24                2              1             GPH2        DSC                  3036                        1
  24                2              1             DSC         FMP                  3037                        0
       JOB #  24 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3038 SEC.. AND HAS SEIZED IT AT REL. CLK. 3038 FOR   10 SECONDS.
                        JOB # 23 COMPLETE AT RELATIVE CLOCK =  3040 SEC.
```

```
                                   FILE TRANSFER SUMMARY - PAGE 16

JOB NO.        FILE NO.        NUMBER OF         FROM       TO        TRANSFER COMPLETED AT         DURATION
                           BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

  24              9              90              FMP       DSC                3053                     3

  94              7               3              SPS       GPH2               3060                     1

                         JOB # 94 COMPLETE AT RELATIVE CLOCK =  3061 SEC.

        JOB #  67 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3077 SEC., AND HAS SEIZED IT AT REL. CLK. 3077 FOR    1 SECONDS.

  67              1               2              SPS       DSC                3080                     1

  24              9              90              DSC       SPS                3081                    26

  67              1               2              DSC       FMP                3081                     0

        JOB #  24 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3082 SEC., AND HAS SEIZED IT AT REL. CLK. 3082 FOR   60 SECONDS.

  67              2               2              GPH2      SPS                3082                     1

        JOB #  67 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3083 SEC., AND HAS SEIZED IT AT REL. CLK. 3083 FOR   80 SECONDS.

        JOB #  18 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3087 SEC., AND HAS SEIZED IT AT REL. CLK. 3087 FOR    1 SECONDS.

  18              1               1              SPS       DSC                3089                     0

  63              7               8              SPS       GPH2               3090                     4

  18              1               1              DSC       FMP                3090                     0

                         JOB # 63 COMPLETE AT RELATIVE CLOCK =  3091 SEC.

  18              2               1              GPH2      DSC                3091                     1

  18              2               1              DSC       FMP                3092                     0

        JOB #  18 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3093 SEC., AND HAS SEIZED IT AT REL. CLK. 3093 FOR   10 SECONDS.

  18              8               2              FMP       DSC                3105                     0

  18              8               2              DSC       SPS                3106                     0

        JOB #  18 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3107 SEC., AND HAS SEIZED IT AT REL. CLK. 3107 FOR   60 SECONDS.

                         JOB # 24 COMPLETE AT RELATIVE CLOCK =  3143 SEC.

  67              2               2              SPS       DSC                3165                     0

  67              2               2              DSC       FMP                3166                     0

                         JOB # 18 COMPLETE AT RELATIVE CLOCK =  3168 SEC.
```

```
                                        FILE TRANSFER SUMMARY - PAGE 17

JOB NO.          FILE NO.       NUMBER OF           FROM       TO         TRANSFER COMPLETED AT           DURATION
                             BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...    OF TRANSFER (SEC.)

  67                 3              19               SPS        DSC                3172                         5

  67                 3              19               DSC        FMP                3174                         0

        JOB #  67 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3175 SEC., AND HAS SEIZED IT AT REL. CLK. 3175 FOR   60 SECONDS.

        JOB #  25 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3208 SEC., AND HAS SEIZED IT AT REL. CLK. 3208 FOR    1 SECONDS.

  95                 1               4               SPS        DSC                3228                         1

  95                 1               4               DSC        FMP                3229                         0

        JOB #  26 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3231 SEC., AND HAS SEIZED IT AT REL. CLK. 3231 FOR    1 SECONDS.

  95                 2               4              GPH1        SPS                3231                         2

        JOB #  95 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3232 SEC., AND HAS SEIZED IT AT REL. CLK. 3232 FOR  360 SECONDS.

  67                 9             125               FMP        DSC                3240                         4

  67                 9             125               FMP        DSC                3247                         5

        JOB #  68 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3280 SEC., AND HAS SEIZED IT AT REL. CLK. 3280 FOR    1 SECONDS.

  67                 9             125               DSC        SPS                3309                        68

  67                 9             125               DSC        SPS                3315                        67

                           JOB # 67 COMPLETE AT RELATIVE CLOCK =  3316 SEC.

        JOB #  69 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3338 SEC., AND HAS SEIZED IT AT REL. CLK. 3338 FOR    1 SECONDS.

        JOB #  65 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3483 SEC., AND HAS SEIZED IT AT REL. CLK. 3483 FOR    1 SECONDS.

  65                 1               2               SPS        DSC                3485                         0

  65                 1               2               DSC        FMP                3487                         0

  65                 2               2              GPH2        DSC                3489                         2

  65                 2               2               DSC        FMP                3491                         0
```

```
                                         FILE TRANSFER SUMMARY - PAGE 18

JOB NO.          FILE NO.       NUMBER OF          FROM       TO        TRANSFER COMPLETED AT           DURATION
                             BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

   65                3              19             SPS        DSC                3496                        5
   65                3              19             DSC        FMP                3498                        0
         JOB #  65 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3499 SEC., AND HAS SEIZED IT AT REL. CLK. 3499 FOR   60 SECONDS.
         JOB #  70 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3530 SEC., AND HAS SEIZED IT AT REL. CLK. 3530 FOR    1 SECONDS.
   65                8               6             FMP        DSC                3560                        0
   65                8               6             DSC        SPS                3563                        2
         JOB #  65 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3564 SEC., AND HAS SEIZED IT AT REL. CLK. 3564 FOR  180 SECONDS.
         JOB #  66 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3566 SEC., AND HAS SEIZED IT AT REL. CLK. 3592 FOR    1 SECONDS.
   66                1               2             SPS        DSC                3595                        0
   66                1               2             DSC        FMP                3596                        0
   66                2               2             GPH1       DSC                3599                        3
   66                2               2             DSC        FMP                3601                        0
   66                3              19             SPS        DSC                3612                       11
   66                3              19             DSC        FMP                3614                        1
         JOB #  66 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3615 SEC., AND HAS SEIZED IT AT REL. CLK. 3615 FOR   60 SECONDS.
   95                3              46             SPS        DSC                3615                       21
   95                3              46             DSC        FMP                3619                        2
   95                3              46             SPS        DSC                3630                       14
   95                3              46             DSC        FMP                3633                        1
         JOB #  95 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3634 SEC., AND HAS SEIZED IT AT REL. CLK. 3675 FOR  600 SECONDS.
   66                9             125             FMP        DSC                3681                        4
   86                1               4             SPS        DSC                3684                        2
   86                1               4             DSC        FMP                3686                        0
   86                2               4             SPS        DSC                3687                        1
```

```
                              FILE TRANSFER SUMMARY - PAGE 19

OB NO.        FILE NO.      NUMBER OF          FROM       TO        TRANSFER COMPLETED AT           DURATION
                         BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

 86              2              4              DSC        FMP                3689                       0

 66              9            125              FMP        DSC                3689                       7

      JOB #  71 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3729 SEC.. AND HAS SEIZED IT AT REL. CLK. 3729 FOR    1 SECONDS.

 65              7              4              SPS        GPH2               3747                       2

                    JOB # 65 COMPLETE AT RELATIVE CLOCK =  3748 SEC.

      JOB #  27 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3750 SEC.. AND HAS SEIZED IT AT REL. CLK. 3750 FOR    1 SECONDS.

 66              9            125              DSC        SPS                3751                      69

 27              1              1              SPS        DSC                3752                       0

 27              1              1              DSC        FMP                3754                       0

 96              1              4              SPS        DSC                3755                       1

 27              2              1              GPH1       DSC                3756                       2

 96              1              4              DSC        FMP                3757                       0

 27              2              1              DSC        FMP                3757                       0

 96              2              4              SPS        DSC                3758                       1

 66              9            125              DSC        SPS                3758                      67

                    JOB # 66 COMPLETE AT RELATIVE CLOCK =  3759 SEC.

 96              2              4              DSC        FMP                3759                       0

      JOB #  72 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3799 SEC.. AND HAS SEIZED IT AT REL. CLK. 3799 FOR    1 SECONDS.

 72              1              2              SPS        DSC                3802                       0

 72              1              2              DSC        FMP                3803                       0

 72              2              2              GPH1       SPS                3804                       1

      JOB #  72 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3805 SEC.. AND HAS SEIZED IT AT REL. CLK. 3805 FOR   80 SECONDS.
```

```
                          FILE TRANSFER SUMMARY - PAGE 20

JOB NO.        FILE NO.       NUMBER OF          FROM       TO        TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

     JOB #  25 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3809 SEC., AND HAS SEIZED IT AT REL. CLK. 3809 FOR    1 SECONDS.
  25              1             1             SPS       DSC                  3811                       0
  25              1             1             DSC       FMP                  3812                       0
  25              2             1             GPH1      DSC                  3814                       1
  25              2             1             DSC       FMP                  3815                       0
     JOB #  26 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3832 SEC., AND HAS SEIZED IT AT REL. CLK. 3832 FOR    1 SECONDS.
     JOB #  68 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3881 SEC., AND HAS SEIZED IT AT REL. CLK. 3881 FOR    1 SECONDS.
  72              2             2             SPS       DSC                  3886                       0
  72              2             2             DSC       FMP                  3888                       0
  72              3            19             SPS       DSC                  3893                       5
  72              3            19             DSC       FMP                  3895                       0
  97              1             4             SPS       DSC                  3927                       1
  97              1             4             DSC       FMP                  3929                       0
  97              2             4             SPS       DSC                  3930                       1
  97              2             4             DSC       FMP                  3931                       0
     JOB #  28 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 3932 SEC., AND HAS SEIZED IT AT REL. CLK. 3932 FOR    1 SECONDS.
     JOB #  69 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 3939 SEC., AND HAS SEIZED IT AT REL. CLK. 3939 FOR    1 SECONDS.
  69              1             2             SPS       DSC                  3942                       0
  69              1             2             DSC       FMP                  3944                       0
  69              2             2             GPH2      DSC                  3946                       2
  69              2             2             DSC       FMP                  3947                       0
  69              3            19             SPS       DSC                  3953                       5
  69              3            19             DSC       FMP                  3955                       0
     JOB #  70 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4131 SEC., AND HAS SEIZED IT AT REL. CLK. 4131 FOR    1 SECONDS.
```

```
                                    FILE TRANSFER SUMMARY - PAGE 21

JOB NO.        FILE NO.       NUMBER OF          FROM       TO        TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

      JOB #  26 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4133 SEC.. AND HAS SEIZED IT AT REL. CLK. 4133 FOR    1 SECONDS.
   70              1              2              SPS        DSC              4134                         0
   26              1              1              SPS        DSC              4135                         0
   70              1              2              DSC        FMP              4135                         0
   26              1              1              DSC        FMP              4137                         0
   26              2              1              GPH2       DSC              4138                         1
   70              2              2              GPH1       DSC              4138                         3
   26              2              1              DSC        FMP              4139                         0
   70              2              2              DSC        FMP              4140                         0
   70              3             19              SPS        DSC              4145                         5
   70              3             19              DSC        FMP              4147                         0
      JOB #  73 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4152 SEC.. AND HAS SEIZED IT AT REL. CLK. 4152 FOR    1 SECONDS.
      JOB #  68 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4182 SEC.. AND HAS SEIZED IT AT REL. CLK. 4182 FOR    1 SECONDS.
   68              1              2              SPS        DSC              4185                         0
   68              1              2              DSC        FMP              4186                         0
   68              2              2              GPH1       DSC              4189                         3
   68              2              2              DSC        FMP              4190                         0
   68              3             19              SPS        DSC              4195                         5
   68              3             19              DSC        FMP              4197                         0
      JOB #  86 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3690 SEC.. AND HAS SEIZED IT AT REL. CLK. 4275 FOR   10 SECONDS.
   95              8              8              FMP        DSC              4277                         0
   95              8              8              DSC        SPS              4280                         2
      JOB #  95 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4281 SEC.. AND HAS SEIZED IT AT REL. CLK. 4281 FOR  360 SECONDS.
      JOB #  27 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3758 SEC.. AND HAS SEIZED IT AT REL. CLK. 4285 FOR   10 SECONDS.
      JOB #  96 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3760 SEC.. AND HAS SEIZED IT AT REL. CLK. 4295 FOR   10 SECONDS.
```

```
                              FILE TRANSFER SUMMARY - PAGE 22

JOB NO.       FILE NO.       NUMBER OF          FROM      TO       TRANSFER COMPLETED AT          DURATION
                         BLOCKS TRANSFERRED                       RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  27              8              2              FMP       DSC                4297                        0

  27              8              2              DSC       SPS                4298                        0

     JOB #  27 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4299 SEC., AND HAS SEIZED IT AT REL. CLK. 4299 FOR   60 SECONDS.

     JOB #  25 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3816 SEC., AND HAS SEIZED IT AT REL. CLK. 4305 FOR   10 SECONDS.

     JOB #  72 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3896 SEC., AND HAS SEIZED IT AT REL. CLK. 4315 FOR   60 SECONDS.

  25              8              2              FMP       DSC                4317                        0

  25              8              2              DSC       SPS                4318                        0

     JOB #  25 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4319 SEC., AND HAS SEIZED IT AT REL. CLK. 4319 FOR   60 SECONDS.

     JOB #  71 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4330 SEC., AND HAS SEIZED IT AT REL. CLK. 4330 FOR    1 SECONDS.

  71              1              2              SPS       DSC                4333                        0

  71              1              2              DSC       FMP                4334                        0

  71              2              2              GPH2      DSC                4336                        2

  71              2              2              DSC       FMP                4338                        0

  71              3             19              SPS       DSC                4343                        5

  71              3             19              DSC       FMP                4345                        0

                        JOB # 27 COMPLETE AT RELATIVE CLOCK =  4360 SEC.

     JOB #  97 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3932 SEC., AND HAS SEIZED IT AT REL. CLK. 4375 FOR   10 SECONDS.

  72              8              6              FMP       DSC                4377                        0

  72              8              6              DSC       SPS                4380                        1

                        JOB # 25 COMPLETE AT RELATIVE CLOCK =  4380 SEC.

     JOB #  72 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4381 SEC., AND HAS SEIZED IT AT REL. CLK. 4381 FOR  180 SECONDS.

     JOB #  69 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 3956 SEC., AND HAS SEIZED IT AT REL. CLK. 4385 FOR   60 SECONDS.

     JOB #  26 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4140 SEC., AND HAS SEIZED IT AT REL. CLK. 4445 FOR   10 SECONDS.

  69              8              6              FMP       DSC                4447                        0

  69              8              6              DSC       SPS                4450                        1

     JOB #  69 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4451 SEC., AND HAS SEIZED IT AT REL. CLK. 4451 FOR  180 SECONDS.
```

```
                              FILE TRANSFER SUMMARY - PAGE 23

JOB NO.       FILE NO.       NUMBER OF          FROM        TO        TRANSFER COMPLETED AT           DURATION
                         BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

       JOB #  70 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4148 SEC.. AND HAS SEIZED IT AT REL. CLK. 4455 FOR   60 SECONDS.
  26               8              2              FMP         DSC                  4457                        0
  26               8              2              DSC         SPS                  4458                        0
       JOB #  26 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4459 SEC.. AND HAS SEIZED IT AT REL. CLK. 4459 FOR   60 SECONDS.
       JOB #  68 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4198 SEC.. AND HAS SEIZED IT AT REL. CLK. 4515 FOR   60 SECONDS.
  70               8             12              FMP         DSC                  4517                        0
                        JOB # 26 COMPLETE AT RELATIVE CLOCK =  4520 SEC.
  70               8             12              DSC         SPS                  4521                        3
       JOB #  70 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4522 SEC.. AND HAS SEIZED IT AT REL. CLK. 4522 FOR  360 SECONDS.
       JOB #  29 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4532 SEC.. AND HAS SEIZED IT AT REL. CLK. 4532 FOR    1 SECONDS.
       JOB #  28 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4533 SEC.. AND HAS SEIZED IT AT REL. CLK. 4533 FOR    1 SECONDS.
  28               1              1              SPS         DSC                  4536                        0
  28               1              1              DSC         FMP                  4537                        0
  28               2              1             GPH1         DSC                  4538                        1
  28               2              1              DSC         FMP                  4540                        0
  72               7              4              SPS        GPH1                  4564                        2
                        JOB # 72 COMPLETE AT RELATIVE CLOCK =  4565 SEC.
       JOB #  86 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4286 SEC.. AND HAS SEIZED IT AT REL. CLK. 4575 FOR   60 SECONDS.
  68               8              6              FMP         DSC                  4577                        0
  87               1              4              SPS         DSC                  4578                        1
  87               1              4              DSC         FMP                  4579                        0
  68               8              6              DSC         SPS                  4580                        2
  87               2              4             GPH2         SPS                  4581                        2
       JOB #  87 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4582 SEC.. AND HAS SEIZED IT AT REL. CLK. 4582 FOR  360 SECONDS.
```

```
                                    FILE TRANSFER SUMMARY - PAGE 24

JOB NO.        FILE NO.       NUMBER OF          FROM      TO        TRANSFER COMPLETED AT          DURATION
                           BLOCKS TRANSFERRED                       RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

    JOB #  68 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4581 SEC., AND HAS SEIZED IT AT REL. CLK. 4631 FOR  180 SECONDS.
  69               7             4               SPS      GPH2                 4634                    2
                         JOB # 69 COMPLETE AT RELATIVE CLOCK =  4635 SEC.
    JOB #  96 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4306 SEC., AND HAS SEIZED IT AT REL. CLK. 4635 FOR  600 SECONDS.
  86               9            160              FMP      DSC                  4642                    5
    JOB #  95 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4642 SEC., AND HAS SEIZED IT AT REL. CLK. 4642 FOR  360 SECONDS.
  86               8             3               FMP      DSC                  4643                    0
  86               9            160              DSC      SPS                  4689                   46
  86               8             3               DSC      SPS                  4690                    0
    JOB #  86 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4691 SEC., AND HAS SEIZED IT AT REL. CLK. 4811 FOR  300 SECONDS.
  68               7             4               SPS      GPH1                 4814                    2
                         JOB # 68 COMPLETE AT RELATIVE CLOCK =  4815 SEC.
  70               7             8               SPS      GPH1                 4888                    4
                         JOB # 70 COMPLETE AT RELATIVE CLOCK =  4889 SEC.
  88               1             4               SPS      DSC                  4890                    1
  88               1             4               DSC      FMP                  4892                    0
  88               2             4               SPS      DSC                  4892                    1
  88               2             4               DSC      FMP                  4894                    0
  98               1             4               SPS      DSC                  4901                    1
  98               1             4               DSC      FMP                  4902                    0
  98               2             4              GPH2      SPS                  4905                    2
    JOB #  98 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 4906 SEC., AND HAS SEIZED IT AT REL. CLK. 4906 FOR  360 SECONDS.
    JOB #  73 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4753 SEC., AND HAS SEIZED IT AT REL. CLK. 4942 FOR    I SECONDS.
  87               3             92              SPS      DSC                  4970                   26
```

```
                                FILE TRANSFER SUMMARY - PAGE 25

JOB NO.        FILE NO.       NUMBER OF          FROM      TO        TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                        RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  87              3              92               DSC       FMP                4974                       3

      JOB #  30 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 4992 SEC., AND HAS SEIZED IT AT REL. CLK. 4992 FOR    1 SECONDS.

  95              7               3               SPS       GPH1               5005                       1

                          JOB # 95 COMPLETE AT RELATIVE CLOCK =  5006 SEC.

      JOB #  31 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5035 SEC., AND HAS SEIZED IT AT REL. CLK. 5035 FOR    1 SECONDS.

  31              1               1               SPS       DSC                5037                       0

      JOB #  32 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5038 SEC., AND HAS SEIZED IT AT REL. CLK. 5038 FOR    1 SECONDS.

  31              1               1               DSC       FMP                5038                       0

  31              2               1               GPH2      DSC                5040                       1

  31              2               1               DSC       FMP                5041                       0

      JOB #  33 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5102 SEC., AND HAS SEIZED IT AT REL. CLK. 5102 FOR    1 SECONDS.

  86              7               2               SPS       GPH2               5113                       1

                          JOB # 86 COMPLETE AT RELATIVE CLOCK =  5114 SEC.

      JOB #  29 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5133 SEC., AND HAS SEIZED IT AT REL. CLK. 5133 FOR    1 SECONDS.

  29              1               1               SPS       DSC                5136                       0

  29              1               1               DSC       FMP                5137                       0

  29              2               1               GPH2      DSC                5138                       1

  29              2               1               DSC       FMP                5139                       0

      JOB #  71 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4346 SEC., AND HAS SEIZED IT AT REL. CLK. 5235 FOR   60 SECONDS.

  96              9             160               FMP       DSC                5242                       5

      JOB #  73 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5243 SEC., AND HAS SEIZED IT AT REL. CLK. 5243 FOR    1 SECONDS.

  96              8               8               FMP       DSC                5244                       0
```

```
                               FILE TRANSFER SUMMARY - PAGE 26

JOB NO.        FILE NO.     NUMBER OF          FROM      TO       TRANSFER COMPLETED AT         DURATION
                         BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

  73              1             2               SPS      DSC             5246                      0
  73              1             2               DSC      FMP             5247                      0
  96              8             8               DSC      SPS             5249                      4
           JOB #  96 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 5250 SEC., AND HAS SEIZED IT AT REL. CLK. 5250 FOR  360 SECONDS.
  73              2             2               GPH1     DSC             5254                      6
  73              2             2               DSC      FMP             5255                      0
           JOB #  34 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5257 SEC., AND HAS SEIZED IT AT REL. CLK. 5257 FOR    1 SECONDS.
  73              3            19               SPS      DSC             5262                      6
  73              3            19               DSC      FMP             5264                      1
  98              3            46               SPS      DSC             5295                     28
           JOB #  97 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4386 SEC., AND HAS SEIZED IT AT REL. CLK. 5295 FOR  600 SECONDS.
  98              3            46               DSC      FMP             5303                      6
  71              9           125               FMP      DSC             5309                     12
  71              9           125               FMP      DSC             5320                      9
  98              3            46               SPS      DSC             5337                     40
  98              3            46               DSC      FMP             5343                      4
  96              9           160               DSC      SPS             5343                     99
           JOB #  35 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5366 SEC., AND HAS SEIZED IT AT REL. CLK. 5366 FOR    1 SECONDS.
  35              1             1               SPS      DSC             5368                      0
  35              1             1               DSC      FMP             5369                      0
  35              2             1               GPH1     DSC             5373                      4
  35              2             1               DSC      FMP             5375                      0
  71              9           125               DSC      SPS             5390                     87
  71              9           125               DSC      SPS             5402                     80
```

```
                                    FILE TRANSFER SUMMARY - PAGE 27

JOB NO.          FILE NO.        NUMBER OF          FROM        TO          TRANSFER COMPLETED AT          DURATION
                            BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

                              JOB # 71 COMPLETE AT RELATIVE CLOCK =  5403 SEC.
        JOB #  36 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5518 SEC., AND HAS SEIZED IT AT REL. CLK. 5518 FOR    1 SECONDS.
        JOB #  37 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5523 SEC., AND HAS SEIZED IT AT REL. CLK. 5523 FOR    1 SECONDS.
        JOB #  38 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5590 SEC., AND HAS SEIZED IT AT REL. CLK. 5590 FOR    1 SECONDS.
        JOB #  30 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5593 SEC., AND HAS SEIZED IT AT REL. CLK. 5593 FOR    1 SECONDS.
        JOB #  96 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 5611 SEC., AND HAS SEIZED IT AT REL. CLK. 5611 FOR  360 SECONDS.
        JOB #  32 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5639 SEC., AND HAS SEIZED IT AT REL. CLK. 5639 FOR    1 SECONDS.
   32                1              1             SPS         DSC                  5641                        0
   32                1              1             DSC         FMP                  5643                        0
   32                2              1             GPH1        DSC                  5644                        1
   32                2              1             DSC         FMP                  5645                        0
        JOB #  39 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5656 SEC., AND HAS SEIZED IT AT REL. CLK. 5656 FOR    1 SECONDS.
   39                1              1             SPS         DSC                  5658                        0
   39                1              1             DSC         FMP                  5659                        0
   39                2              1             GPH1        DSC                  5661                        1
   39                2              1             DSC         FMP                  5662                        0
        JOB #  40 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5690 SEC., AND HAS SEIZED IT AT REL. CLK. 5690 FOR    1 SECONDS.
        JOB #  33 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5703 SEC., AND HAS SEIZED IT AT REL. CLK. 5703 FOR    1 SECONDS.
   33                1              1             SPS         DSC                  5705                        0
   33                1              1             DSC         FMP                  5706                        0
   33                2              1             GPH1        DSC                  5708                        1
```

```
                         FILE TRANSFER SUMMARY - PAGE 28

JOB NO.      FILE NO.      NUMBER OF         FROM      TO       TRANSFER COMPLETED AT       DURATION
                        BLOCKS TRANSFERRED                     RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

  33            2             1              DSC       FMP              5709                   0
      JOB #  34 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5858 SEC.. AND HAS SEIZED IT AT REL. CLK. 5858 FOR    1 SECONDS.
      JOB #  74 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5879 SEC.. AND HAS SEIZED IT AT REL. CLK. 5879 FOR    1 SECONDS.
      JOB #  41 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5880 SEC.. AND HAS SEIZED IT AT REL. CLK. 5880 FOR    1 SECONDS.
  41            1             1              SPS       DSC              5882                   0
  41            1             1              DSC       FMP              5883                   0
  41            2             1              GPH2      DSC              5884                   1
  41            2             1              DSC       FMP              5886                   0
      JOB #  30 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5894 SEC.. AND HAS SEIZED IT AT REL. CLK. 5894 FOR    1 SECONDS.
      JOB #  28 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4541 SEC.. AND HAS SEIZED IT AT REL. CLK. 5895 FOR   10 SECONDS.
  30            1             1              SPS       DSC              5897                   0
  30            1             1              DSC       FMP              5898                   0
  30            2             1              GPH1      DSC              5899                   1
  30            2             1              DSC       FMP              5901                   0
      JOB #  88 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4895 SEC.. AND HAS SEIZED IT AT REL. CLK. 5905 FOR   10 SECONDS.
  97            9           310              FMP       DSC              5911                  14
  97            8             8              FMP       DSC              5912                   0
  28            9            90              FMP       DSC              5913                   6
      JOB #  87 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 4975 SEC.. AND HAS SEIZED IT AT REL. CLK. 5915 FOR   60 SECONDS.
  97            8             8              DSC       SPS              5918                   4
      JOB #  97 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5919 SEC.. AND HAS SEIZED IT AT REL. CLK. 5919 FOR  360 SECONDS.
  28            9            90              DSC       SPS              5942                  28
      JOB #  28 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5943 SEC.. AND HAS SEIZED IT AT REL. CLK. 5943 FOR   60 SECONDS.
```

```
                            FILE TRANSFER SUMMARY - PAGE 29

JOB NO.        FILE NO.      NUMBER OF          FROM      TO       TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...  OF TRANSFER (SEC.)

  99              1              4              SPS       DSC              5946                    1

  99              1              4              DSC       FMP              5947                    0

  99              2              4              SPS       DSC              5948                    1

  99              2              4              DSC       FMP              5949                    0

  96              7              3              SPS       GPH2             5974                    1

                    JOB # 96 COMPLETE AT RELATIVE CLOCK =  5975 SEC.

      JOB #  31 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5042 SEC.. AND HAS SEIZED IT AT REL. CLK. 5975 FOR   10 SECONDS.

  87              8              3              FMP       DSC              5977                    0

  87              8              3              DSC       SPS              5979                    1

      JOB #  87 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 5980 SEC.. AND HAS SEIZED IT AT REL. CLK. 5980 FOR  300 SECONDS.

      JOB #  29 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5140 SEC.. AND HAS SEIZED IT AT REL. CLK. 5985 FOR   10 SECONDS.

  31              8              2              FMP       DSC              5987                    0

      JOB #  73 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5265 SEC.. AND HAS SEIZED IT AT REL. CLK. 5995 FOR   60 SECONDS.

  29              8              2              FMP       DSC              5997                    0

  29              8              2              DSC       SPS              6004                    0

                    JOB # 28 COMPLETE AT RELATIVE CLOCK =  6004 SEC.

  31              8              2              DSC       SPS              6004                    1

      JOB #  29 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6005 SEC.. AND HAS SEIZED IT AT REL. CLK. 6005 FOR   60 SECONDS.

      JOB #  98 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5344 SEC.. AND HAS SEIZED IT AT REL. CLK. 6055 FOR  600 SECONDS.

  73              8              6              FMP       DSC              6057                    0

      JOB #  31 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6005 SEC.. AND HAS SEIZED IT AT REL. CLK. 6065 FOR   60 SECONDS.

                    JOB # 29 COMPLETE AT RELATIVE CLOCK =  6066 SEC.

      JOB #  42 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6078 SEC.. AND HAS SEIZED IT AT REL. CLK. 6078 FOR    1 SECONDS.

  42              1              1              SPS       DSC              6080                    0

  42              1              1              DSC       FMP              6082                    0
```

```
                                     FILE TRANSFER SUMMARY - PAGE 30

JOB NO.         FILE NO.        NUMBER OF         FROM        TO        TRANSFER COMPLETED AT          DURATION
                            BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

   42              2                1             GPH1        DSC               6083                       1

   42              2                1              DSC        FMP               6084                       0

       JOB #  43 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6088 SEC., AND HAS SEIZED IT AT REL. CLK. 6088 FOR    1 SECONDS.

   43              1                1              SPS        DSC               6090                       0

   43              1                1              DSC        FMP               6092                       0

   43              2                1             GPH2        DSC               6093                       1

   43              2                1              DSC        FMP               6094                       0

                        JOB # 31 COMPLETE AT RELATIVE CLOCK =  6126 SEC.

   73              8                6              DSC        SPS               6127                       1

       JOB #  36 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6119 SEC., AND HAS SEIZED IT AT REL. CLK. 6127 FOR    1 SECONDS.

       JOB #  37 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6124 SEC., AND HAS SEIZED IT AT REL. CLK. 6128 FOR    1 SECONDS.

   36              1                1              SPS        DSC               6129                       0

   37              1                1              SPS        DSC               6130                       0

   36              1                1              DSC        FMP               6131                       0

   37              1                1              DSC        FMP               6132                       0

   36              2                1             GPH2        DSC               6132                       1

   37              2                1             GPH1        DSC               6133                       1

   36              2                1              DSC        FMP               6133                       0

   37              2                1              DSC        FMP               6134                       0

       JOB #  75 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6148 SEC., AND HAS SEIZED IT AT REL. CLK. 6148 FOR    1 SECONDS.

       JOB #  34 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6159 SEC., AND HAS SEIZED IT AT REL. CLK. 6159 FOR    1 SECONDS.

   34              1                1              SPS        DSC               6161                       0

   34              1                1              DSC        FMP               6163                       0

   34              2                1             GPH2        DSC               6164                       1
```

```
                                        FILE TRANSFER SUMMARY - PAGE 31

JOB NO.         FILE NO.      NUMBER OF          FROM       TO          TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  34               2             1               DSC        FMP                  6165                        0
       JOB #  38 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6191 SEC.. AND HAS SEIZED IT AT REL. CLK. 6191 FOR    1 SECONDS.
       JOB #  76 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6215 SEC.. AND HAS SEIZED IT AT REL. CLK. 6215 FOR    1 SECONDS.
       JOB #  73 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6128 SEC.. AND HAS SEIZED IT AT REL. CLK. 6279 FOR  180 SECONDS.
       JOB #  97 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6280 SEC.. AND HAS SEIZED IT AT REL. CLK. 6280 FOR  360 SECONDS.
  87               7             2               SPS        GPH2                 6282                        1
                          JOB # 87 COMPLETE AT RELATIVE CLOCK =  6283 SEC.
       JOB #  40 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6291 SEC.. AND HAS SEIZED IT AT REL. CLK. 6291 FOR    1 SECONDS.
  40               1             1               SPS        DSC                  6294                        0
  40               1             1               DSC        FMP                  6295                        0
  40               2             1               GPH2       DSC                  6296                        1
  40               2             1               DSC        FMP                  6297                        0
       JOB #  44 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6357 SEC.. AND HAS SEIZED IT AT REL. CLK. 6357 FOR    1 SECONDS.
  44               1             1               SPS        DSC                  6359                        0
  44               1             1               DSC        FMP                  6361                        0
  44               2             1               GPH1       DSC                  6362                        1
  44               2             1               DSC        FMP                  6363                        0
  73               7             4               SPS        GPH1                 6462                        2
                          JOB # 73 COMPLETE AT RELATIVE CLOCK =  6463 SEC.
       JOB #  74 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6480 SEC.. AND HAS SEIZED IT AT REL. CLK. 6480 FOR    1 SECONDS.
  74               1             2               SPS        DSC                  6482                        0
  74               1             2               DSC        FMP                  6483                        0
  74               2             2               GPH1       DSC                  6486                        2
```

```
                         FILE TRANSFER SUMMARY - PAGE 32

JOB NO.         FILE NO.        NUMBER OF          FROM       TO        TRANSFER COMPLETED AT          DURATION
                            BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

        JOB #  77 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6486 SEC., AND HAS SEIZED IT AT REL. CLK. 6486 FOR    1 SECONDS.
  74                2               2              DSC        FMP              6487                          0
  77                1               2              SPS        DSC              6489                          0
  77                1               2              DSC        FMP              6490                          0
  77                2               2              GPH1       SPS              6491                          1
        JOB #  38 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6492 SEC., AND HAS SEIZED IT AT REL. CLK. 6492 FOR    1 SECONDS.
        JOB #  77 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6492 SEC., AND HAS SEIZED IT AT REL. CLK. 6492 FOR   80 SECONDS.
  74                3              19              SPS        DSC              6493                          6
  38                1               1              SPS        DSC              6494                          0
  74                3              19              DSC        FMP              6495                          0
  38                1               1              DSC        FMP              6495                          0
  38                2               1              GPH1       DSC              6497                          1
  38                2               1              DSC        FMP              6498                          0
        JOB #  78 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6522 SEC., AND HAS SEIZED IT AT REL. CLK. 6522 FOR    1 SECONDS.
        JOB #  45 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6530 SEC., AND HAS SEIZED IT AT REL. CLK. 6530 FOR    1 SECONDS.
  77                2               2              SPS        DSC              6574                          0
  77                2               2              DSC        FMP              6575                          0
  77                3              19              SPS        DSC              6581                          5
  77                3              19              DSC        FMP              6583                          0
  97                7               3              SPS        GPH2             6643                          1
                          JOB # 97 COMPLETE AT RELATIVE CLOCK =  6644 SEC.
        JOB #  35 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5376 SEC., AND HAS SEIZED IT AT REL. CLK. 6655 FOR   10 SECONDS.
  98                8               8              FMP        DSC              6657                          0
  98                8               8              DSC        SPS              6660                          2
```

```
                              FILE TRANSFER SUMMARY - PAGE  3                                DURATION
                                                                                        OF TRANSFER (SEC.)
 JOB NO.        FILE NO.       NUMBER OF          FROM       TO        TRANSFER COMPLETED AT
                           BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC.)    [illegible] FOR  360 SECONDS.
                                                                                                    [illegible] FOR   10 SECONDS.
        JOB  98 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6061 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            0
        JOB  32 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5146 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            0
   31            8              2              FMP        DSC                      5667
                                                                                                    [illegible] FOR   60 SECONDS.
   3             8              2              DSC        SPS                      5668
                                                                                                    [illegible] FOR   10 SECONDS.
        JOB  35 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6069 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            3
        JOB  39 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5163 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                    [illegible] FOR   10 SECONDS.
   32            9             90              FMP        DSC                      5680
                                                                                                            4
        JOB  33 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5 10 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                    [illegible] FOR   10 SECONDS.
   3             9             90              FMP        DSC                      5691
                                                                                                            0
        JOB  31 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5087 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            2
   3             8              2              FMP        DSC                      5697
                                                                                                    [illegible] FOR   60 SECONDS.
   3             8              2              DSC        SPS                      5700
                                                                                                    [illegible] FOR   10 SECONDS.
        JOB  33 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6 01 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            0
        JOB  30 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5102 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            1
   4             8              2              FMP        DSC                      5707
                                                                                                    [illegible] FOR   60 SECONDS.
   4             6              2              DSC        SPS                      5709
                                                                                                    [illegible] FOR   60 SECONDS.
        JOB  31 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6 10 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                            6
        JOB  38 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5 16 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                           69
   3             9             90              FMP        DSC                      5723
   3             9             90              DSC        SPS                      5730
                                                                                                    [illegible] FOR   60 SECONDS.
                           JOB  35 COMPLETE AT RELATIVE CLOCK =  673  SEC.
                                                                                                           57
        JOB  32 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6 31 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                                                                                                    [illegible] FOR    1 SECONDS.
   3             9             90              DSC        SPS                      5749
                                                                                                    [illegible] FOR   60 SECONDS.
        JOB  75 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6 49 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
        JOB  39 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6 50 SEC. AND HAS SEIZED IT AT REL. CLK. [illegible]
                           JOB  33 COMPLETE AT RELATIVE CLOCK =  676  SEC.
```

```
                                   FILE TRANSFER SUMMARY - PAGE 34

JOB NO.        FILE NO.       NUMBER OF          FROM       TO        TRANSFER COMPLETED AT         DURATION
                           BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  30              9             90               DSC        SPS                 6765                    40

     JOB #  30 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6766 SEC.. AND HAS SEIZED IT AT REL. CLK. 6766 FOR   60 SECONDS.

     JOB #  79 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6767 SEC.. AND HAS SEIZED IT AT REL. CLK. 6767 FOR    1 SECONDS.

                       JOB # 41 COMPLETE AT RELATIVE CLOCK =  6771 SEC.

     JOB #  99 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 5950 SEC.. AND HAS SEIZED IT AT REL. CLK. 6775 FOR   10 SECONDS.

  88              9            220               FMP        DSC                 6784                     7

  88              8              3               FMP        DSC                 6785                     0

     JOB #  42 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6085 SEC.. AND HAS SEIZED IT AT REL. CLK. 6785 FOR   10 SECONDS.

  88              8              3               DSC        SPS                 6787                     1

     JOB #  88 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6788 SEC.. AND HAS SEIZED IT AT REL. CLK. 6788 FOR  300 SECONDS.

                       JOB # 32 COMPLETE AT RELATIVE CLOCK =  6792 SEC.

     JOB #  43 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6095 SEC.. AND HAS SEIZED IT AT REL. CLK. 6795 FOR   10 SECONDS.

  42              8              2               FMP        DSC                 6797                     0

  42              8              2               DSC        SPS                 6798                     0

     JOB #  42 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6799 SEC.. AND HAS SEIZED IT AT REL. CLK. 6799 FOR   60 SECONDS.

     JOB #  36 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6134 SEC.. AND HAS SEIZED IT AT REL. CLK. 6805 FOR   10 SECONDS.

  43              9             90               FMP        DSC                 6810                     3

                       JOB # 39 COMPLETE AT RELATIVE CLOCK =  6811 SEC.

     JOB #  37 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6135 SEC.. AND HAS SEIZED IT AT REL. CLK. 6815 FOR   10 SECONDS.

  36              9             90               FMP        DSC                 6820                     3

     JOB #  34 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6166 SEC.. AND HAS SEIZED IT AT REL. CLK. 6825 FOR   10 SECONDS.

  37              8              2               FMP        DSC                 6827                     0

                       JOB # 30 COMPLETE AT RELATIVE CLOCK =  6827 SEC.

  37              8              2               DSC        SPS                 6829                     1

     JOB #  37 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6830 SEC.. AND HAS SEIZED IT AT REL. CLK. 6830 FOR   60 SECONDS.
```

11-A-113

```
                                    FILE TRANSFER SUMMARY - PAGE 35

JOB NO.         FILE NO.        NUMBER OF          FROM       TO         TRANSFER COMPLETED AT          DURATION
                             BLOCKS TRANSFERRED                         RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

        JOB #  40 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6298 SEC.. AND HAS SEIZED IT AT REL. CLK. 6835 FOR   10 SECONDS.
  34                8              2                FMP        DSC                    6837                          0
  34                8              2                DSC        SPS                    6839                          1
        JOB #  34 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6840 SEC.. AND HAS SEIZED IT AT REL. CLK. 6840 FOR   60 SECONDS.
        JOB #  44 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6364 SEC.. AND HAS SEIZED IT AT REL. CLK. 6845 FOR   10 SECONDS.
  36                9             90                DSC        SPS                    6849                         28
        JOB #  36 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6850 SEC.. AND HAS SEIZED IT AT REL. CLK. 6850 FOR   60 SECONDS.
  40                9             90                FMP        DSC                    6850                          4
        JOB #  74 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6496 SEC.. AND HAS SEIZED IT AT REL. CLK. 6855 FOR   60 SECONDS.
                          JOB # 42 COMPLETE AT RELATIVE CLOCK =  6860 SEC.
  44                9             90                FMP        DSC                    6862                          5
                          JOB # 37 COMPLETE AT RELATIVE CLOCK =  6891 SEC.
  40                9             90                DSC        SPS                    6896                         44
        JOB #  40 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6897 SEC.. AND HAS SEIZED IT AT REL. CLK. 6897 FOR   60 SECONDS.
                          JOB # 34 COMPLETE AT RELATIVE CLOCK =  6901 SEC.
                          JOB # 36 COMPLETE AT RELATIVE CLOCK =  6911 SEC.
  43                9             90                DSC        SPS                    6913                         53
  44                9             90                DSC        SPS                    6915                         52
        JOB #  43 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 6914 SEC.. AND HAS SEIZED IT AT REL. CLK. 6915 FOR   60 SECONDS.
        JOB #  38 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6499 SEC.. AND HAS SEIZED IT AT REL. CLK. 6915 FOR   10 SECONDS.
  74                8             12                FMP        DSC                    6917                          0
  74                8             12                DSC        SPS                    6922                          3
        JOB #  74 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6923 SEC.. AND HAS SEIZED IT AT REL. CLK. 6923 FOR  360 SECONDS.
        JOB #  77 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 6584 SEC.. AND HAS SEIZED IT AT REL. CLK. 6925 FOR   60 SECONDS.
        JOB #  46 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 6926 SEC.. AND HAS SEIZED IT AT REL. CLK. 6926 FOR    1 SECONDS.
```

```
                                        FILE TRANSFER SUMMARY - PAGE 36

                            NUMBER OF             FROM        TO        TRANSFER COMPLETED AT          DURATION
                        BLOCKS TRANSFERRED                              RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

JOB NO.         FILE NO.           2               FMP         DSC               6927                      0

                                   2               DSC         SPS               6928                      0
   38             8
                        THE SPS1 PROCESSOR AT REL. CLK. 6929 SEC., AND HAS SEIZED IT AT REL. CLK. 6929 FOR   60 SECONDS.
   38             8
                         JOB # 40 COMPLETE AT RELATIVE CLOCK =  6958 SEC.
       JOB #  33 REQUESTED THE
                         THE SPS2 PROCESSOR AT REL. CLK. 6916 SEC., AND HAS SEIZED IT AT REL. CLK. 6975 FOR   60 SECONDS.
                        JOB
                         JOB # 43 COMPLETE AT RELATIVE CLOCK =  6976 SEC.
       JOB #  46 REQUESTED THE
                        THE  FMP PROCESSOR AT REL. CLK. 6786 SEC., AND HAS SEIZED IT AT RFL. CLK. 6985 FOR  600 SECONDS.
                        JOB
                                   6               FMP         DSC               6987                      0
       JOB #  99 REQUESTED THE
                        JOB # 38 COMPLETE AT RELATIVE CLOCK =  6990 SEC.
   77             8
                        THE SPS2 PROCESSOR AT REL. CLK. 6816 SEC., AND HAS SEIZED IT AT REL. CLK. 7021 FOR    1 SECONDS.
                        JOB
                                   6               DSC         SPS               7023                      2
       JOB #  76 REQUESTED THE
                                   3               SPS         DSC               7024                      0
   77             8
                        THE SPS2 PROCESSOR AT REL. CLK. 7022 SEC., AND HAS SEIZED IT AT REL. CLK. 7024 FOR  360 SECONDS.
   76             1
                                   2               DSC         FMP               7028                      0
       JOB #  98 REQUESTED THE
                                   3               GPH1        DSC               7028                      2
   76             1
                                   2               DSC         FMP               7029                      0
   76             2
                        JOB # 46 COMPLETE AT RELATIVE CLOCK =  7036 SEC.
   76             2
                                  19               SPS         DSC               7041                      5
                        JOB
                                  19               DSC         FMP               7042                      0
   76             3
                        THE SPS2 PROCESSOR AT REL. CLK. 7024 SEC., AND HAS SEIZED IT AT REL. CLK. 7088 FOR  180 SECONDS.
   76             3
                        THE SPS1 PROCESSOR AT REL. CLK. 7089 SEC., AND HAS SEIZED IT AT RFL. CLK. 7089 FOR    1 SECONDS.
       JOB #  77 REQUESTED THE
                                   2               SPS         GPH1              7091                      1
       JOB #  47 REQUESTED THE
                        JOB # 88 COMPLETE AT RELATIVE CLOCK =  7092 SEC.
   88             7
                                   1               SPS         DSC               7092                      0
                        JOB
                                   1               DSC         FMP               7093                      0
   47             1

   47             1
```

```
                              FILE TRANSFER SUMMARY - PAGE 37

JOB NO.        FILE NO.      NUMBER OF          FROM       TO      TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  47              2              1               GPH1       DSC            7094                        1

  47              2              1                DSC       FMP            7095                        0

         JOB #  45 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7131 SEC.. AND HAS SEIZED IT AT REL. CLK. 7131 FOR    1 SECONDS.

  45              1              1                SPS       DSC            7133                        0

  45              1              1                DSC       FMP            7134                        0

  45              2              1               GPH1       DSC            7136                        1

  45              2              1                DSC       FMP            7137                        0

         JOB #  48 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7148 SEC.. AND HAS SEIZED IT AT REL. CLK. 7148 FOR    1 SECONDS.

         JOB #  75 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7050 SEC.. AND HAS SEIZED IT AT REL. CLK. 7268 FOR    1 SECONDS.

         JOB #  78 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7123 SEC.. AND HAS SEIZED IT AT REL. CLK. 7269 FOR    1 SECONDS.

  75              1              2                SPS       DSC            7271                        0

  77              7              4                SPS       GPH1           7272                        2

  75              1              2                DSC       FMP            7273                        0

                     JOB # 77 COMPLETE AT RELATIVE CLOCK =  7273 SEC.

  75              2              2               GPH1       DSC            7275                        2

  75              2              2                DSC       FMP            7276                        0

  75              3             19                SPS       DSC            7282                        5

  75              3             19                DSC       FMP            7283                        0

  74              7              8                SPS       GPH1           7289                        4

                     JOB # 74 COMPLETE AT RELATIVE CLOCK =  7290 SEC.

         JOB #  80 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7302 SEC.. AND HAS SEIZED IT AT REL. CLK. 7302 FOR    1 SECONDS.

         JOB #  49 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7347 SEC.. AND HAS SEIZED IT AT REL. CLK. 7347 FOR    1 SECONDS.
```

```
                              FILE TRANSFER SUMMARY - PAGE 38

JOB NO.        FILE NO.      NUMBER OF        FROM      TO        TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                      RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

          JOB #  50 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7354 SEC.. AND HAS SEIZED IT AT REL. CLK. 7354 FOR    1 SECONDS.

          JOB #  89 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7357 SEC.. AND HAS SEIZED IT AT REL. CLK. 7357 FOR    3 SECONDS.

          JOB #  51 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7358 SEC.. AND HAS SEIZED IT AT REL. CLK. 7358 FOR    1 SECONDS.

  51              1             1             SPS       DSC              7360                        0

  51              1             1             DSC       FMP              7361                        0

  51              2             1             GPH1      DSC              7363                        1

  89              1             4             SPS       DSC              7363                        1

  51              2             1             DSC       FMP              7364                        0

  89              1             4             DSC       FMP              7364                        0

  89              2             4             SPS       DSC              7366                        1

  89              2             4             DSC       FMP              7367                        0

          JOB #  79 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7368 SEC.. AND HAS SEIZED IT AT REL. CLK. 7368 FOR    1 SECONDS.

  79              1             2             SPS       DSC              7371                        0

  79              1             2             DSC       FMP              7372                        0

  79              2             2             GPH1      DSC              7375                        2

  79              2             2             DSC       FMP              7376                        0

  79              3            19             SPS       DSC              7382                        5

  79              3            19             DSC       FMP              7383                        0

  98              7             3             SPS       GPH2             7387                        1

                     JOB # 98 COMPLETE AT RELATIVE CLOCK =  7388 SEC.

          JOB #  46 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7527 SEC.. AND HAS SEIZED IT AT REL. CLK. 7527 FOR    1 SECONDS.

          JOB #  52 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7535 SEC.. AND HAS SEIZED IT AT REL. CLK. 7535 FOR    1 SECONDS.

  52              1             1             SPS       DSC              7537                        0
```

```
                                        FILE TRANSFER SUMMARY - PAGE 39

JOB NO.        FILE NO.      NUMBER OF          FROM       TO        TRANSFER COMPLETED AT          DURATION
                          BLOCKS TRANSFERRED                       RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

  52              1              1               DSC       FMP               7538                     0

  52              2              1               GPH1      DSC               7540                     1

  52              2              1               DSC       FMP               7541                     0

      JOB #  78 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7570 SEC., AND HAS SEIZED IT AT REL. CLK. 7570 FOR    1 SECONDS.

  78              1              2               SPS       DSC               7573                     0

  78              1              2               DSC       FMP               7574                     0

  78              2              2               GPH1      DSC               7577                     2

  78              2              2               DSC       FMP               7578                     0

  78              3             19               SPS       DSC               7584                     5

      JOB #  76 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7043 SEC., AND HAS SEIZED IT AT REL. CLK. 7585 FOR   60 SECONDS.

  78              3             19               DSC       FMP               7586                     0

  99              8              8               FMP       DSC               7587                     0

  99              8              8               DSC       SPS               7590                     2

      JOB #  99 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7591 SEC., AND HAS SEIZED IT AT REL. CLK. 7591 FOR  360 SECONDS.

      JOB #  81 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7595 SEC., AND HAS SEIZED IT AT REL. CLK. 7595 FOR    1 SECONDS.

      JOB #  47 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7096 SEC., AND HAS SEIZED IT AT REL. CLK. 7645 FOR   10 SECONDS.

  76              9            125               FMP       DSC               7651                     4

      JOB #  45 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7138 SEC., AND HAS SEIZED IT AT REL. CLK. 7655 FOR   10 SECONDS.

  47              8              2               FMP       DSC               7657                     0

  76              9            125               FMP       DSC               7658                     6

  47              8              2               DSC       SPS               7659                     0

      JOB #  47 REQUESTED THE SPS1 PROCESSOR AT REL. CLK. 7660 SEC., AND HAS SEIZED IT AT REL. CLK. 7660 FOR   60 SECONDS.

      JOB #  75 REQUESTED THE  FMP PROCESSOR AT REL. CLK. 7284 SEC., AND HAS SEIZED IT AT REL. CLK. 7665 FOR   60 SECONDS.

  45              8              2               FMP       DSC               7667                     0

  45              8              2               DSC       SPS               7670                     2
```

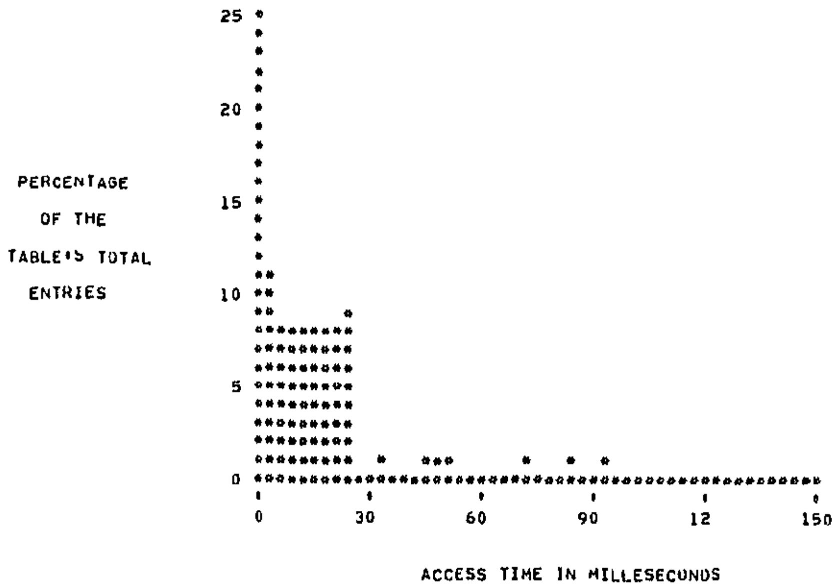
```
                                 FILE TRANSFER SUMMARY - PAGE 40

JOB NO.        FILE NO.        NUMBER OF          FROM       TO        TRANSFER COMPLETED AT          DURATION
                           BLOCKS TRANSFERRED                          RELATIVE CLOCK TIME(SEC)...   OF TRANSFER (SEC.)

    JOB #  45 REQUESTED THE SPS2 PROCESSOR AT REL. CLK. 7671 SEC., AND HAS SEIZED IT AT REL. CLK. 7671 FOR   60 SECONDS.
```

```
                              SIMULATION COMPLETE.  SUMMARY STATISTICS --


THE FOLLOWING STATISTICS SUMMARIZE THE UTILIZATION OF THE PROPOSED NASF HARDWARE.
   ( THE TIME UNIT IS 10-MICROSECONDS UNLESS OTHERWISE NOTED ):


  QUEUING STATISTICS  FOR THE  FMP AND SPS'S

QUEUE      MAXIMUM    AVERAGE     TOTAL      ZERO     PERCENT    AVERAGE      $AVERAGE
           CONTENTS   CONTENTS    ENTRIES    ENTRIES   ZEROS     TIME/TRAN    TIME/TRAN

FMPQ          18       5.265        86         13       15.1 47169699.837 56341585.917
SPS1Q          2       0.067       123        113       91.9   417593.496  5136400.000
SPS2Q          3       0.154        95         78       82.1  1246734.474  6967045.588


          UTILIZATION OF THE FMP

FACILITY      AVERAGE       NUMBER      AVERAGE
            UTILIZATION     ENTRIES     TIME/TRAN
               0.859                    8140479.2
  FMP         ~~0.458~~        81      ~~610125.827~~


       UTILIZATION OF THE SPS AND GRF DEVICES
           ( CONTENTS ARE PERCENTAGES )

STORAGE      AVERAGE       ENTRIES      AVERAGE       CURRENT      MAXIMUM
           UTILIZATION               TIME/TRAN     CONTENTS     CONTENTS

  SPS1         0.469         3508   10294041.105         32          100
  SPS2         0.454         3049   11472069.764         82          100
  GRF1         0.054         1535    2726690.160          5           35
  GRF2         0.053         1295    3173179.564          5           35
```

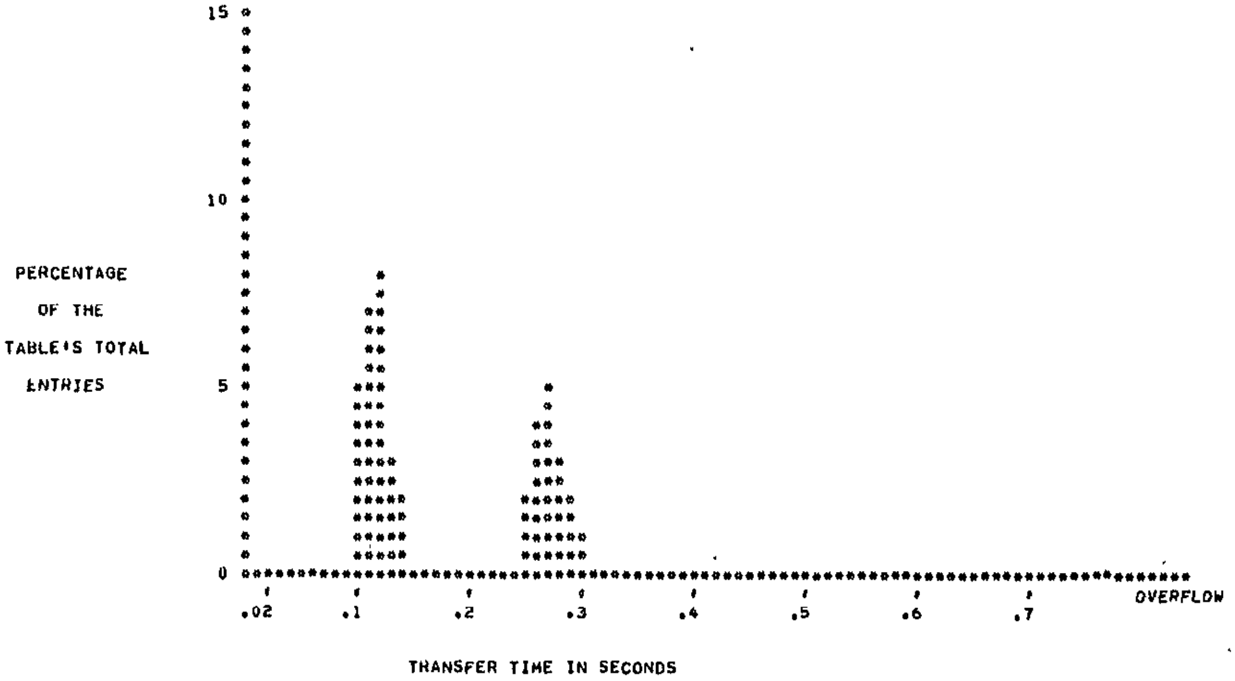
```
ARDEN HILLS CYBER 175 GPSS V/6000            CRM GPSS V/6000   VER. 1.2 PSR 453      03/17/79  15.03.53.   PAGE    350




          UTILIZATION OF THE BIT SERIAL DATA TRUNKS

     FACILITY        AVERAGE         NUMBER       AVERAGE
                   UTILIZATION      ENTRIES      TIME/TRAN

           1           0.021          2907        5593.985
           2           0.021          2901        5580.226
           3           0.021          2930        5530.615
           4           0.021          2929        5573.453
           5           0.006           797        5929.868
           6           0.093          5633       12781.175
           7           0.069          4149       12752.913




                MESSAGE TURNAROUND STATISTICS FOR EACH TRUNK

     QUEUE        MAXIMUM      AVERAGE       TOTAL       AVERAGE
                 CONTENTS     CONTENTS      ENTRIES     TIME/TRAN

         1              2        0.026         2658      7591.124
         2              2        0.026         2630      7619.799
         3              2        0.026         2660      7447.933
         4              2        0.027         2666      7682.457
         5              2        0.006          750      6321.996
         6              3        0.130         5152     19513.780
         7              4        0.106         3781     21659.589
```

```
                 UTILIZATION OF THE PDC UNITS

     STORAGE       AVERAGE         AVERAGE         ENTRIES       AVERAGE
                   CONTENTS      UTILIZATION                    TIME/TRAN

           1         0.000           0.000            286          1.934
           4         0.021           0.021           2907       5593.985
           5         0.021           0.021           2901       5580.226
           6         0.021           0.021           2930       5530.615
           7         0.021           0.021           2929       5573.453
          13         0.003           0.003            409       5801.885
          14         0.089           0.089           5348      12782.166
          17         0.003           0.003            388       6064.778
          18         0.062           0.062           3753      12771.041
          22         0.021           0.021           2907       5593.985
          23         0.041           0.041           2447      12753.395
          24         0.021           0.021           2901       5580.226
          25         0.040           0.040           2432      12818.886
          26         0.021           0.021           2930       5530.615
          27         0.041           0.041           2469      12739.771
          28         0.021           0.021           2929       5573.453
          29         0.040           0.040           2482      12518.375
          37         0.010           0.010            656      11249.959
          38         0.008           0.008            542      11005.589
```

HISTOGRAM OF DISC ACCESS TIMES



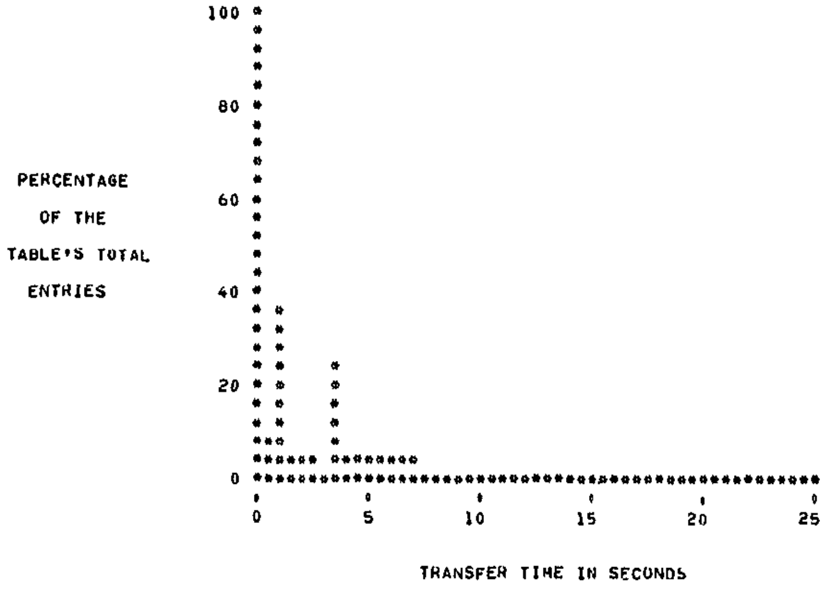
HISTOGRAM OF BLOCK TRANSFER TIMES



NOTE: ABOUT 50 PERCENT OF THIS TABLE'S ENTRIES ARE CONTAINED IN THE FIRST BIN (.LT. .02 SEC). THEY ARE THE SHORT MESSAGE REQUESTS AND RESPONSES AND ARE THEREFORE LEFT OFF THIS GRAPH. CONSEQUENTLY, THE 'Y'-AXIS SHOULD BE READ AS TWICE THE NOTED VALUE.

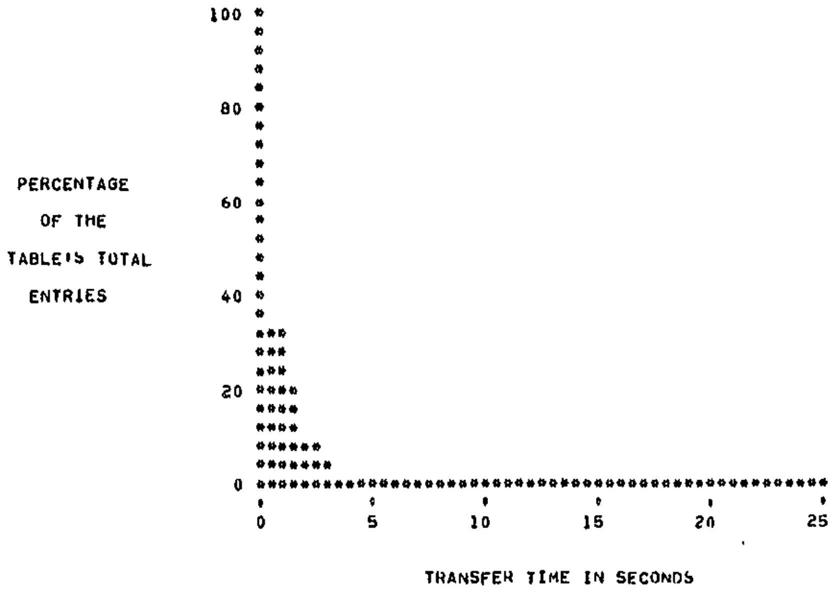
```
                                                 HISTOGRAM OF TRANSFER TIMES FOR SMALL
                                                   FILES ( LESS THAN 10 BLOCKS )
                                                    BETWEEN THE FMP AND DSC

                                     100 *
                                         *
                                         *
                                         *
                                         *
                                      80 *
                                         *
                                         **
                                         **
                   PERCENTAGE            **
                                      60 **
                     OF THE              **
                                         **
                  TABLE'S TOTAL          **
                                         **
                    ENTRIES           40 **
                                         **
                                         **
                                         **
                                         **
                                      20 **
                                         ***
                                         ****
                                         ****
                                         ****
                                       0 ***************************************************
                                         '   '   '   '   '   '   '   '   '   '   '   '   '
                                         0       2       4       6       8      10      12

                                                       TRANSFER TIME IN SECONDS
```

HISTOGRAM OF TRANSFER TIMES FOR INTERMEDIATE
SIZE FILES (MORE THAN 9 BLOCKS, LESS THAN 100)
BETWEEN THE FMP ANC DSC

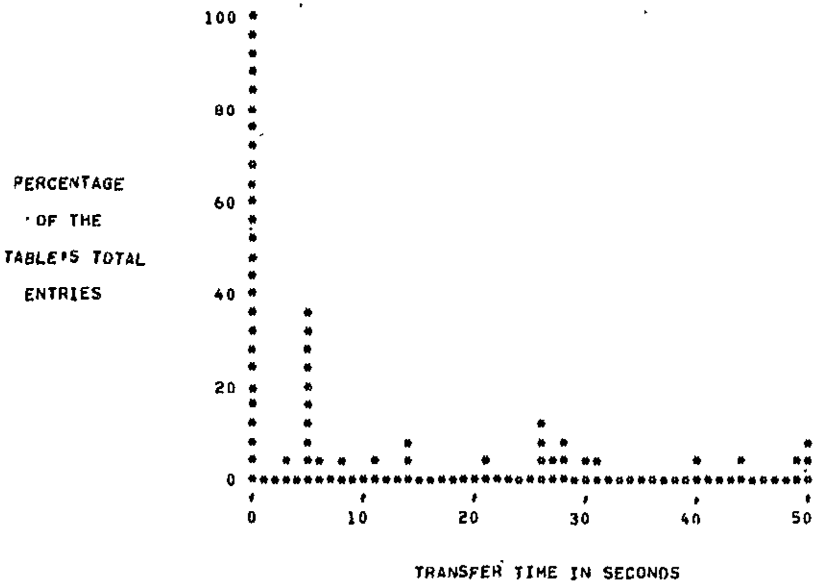


```
                                 HISTOGRAM OF TRANSFER TIMES FOR ALL FILES
                                       BETWEEN THE FMP AND DSC

                   100 *
                       *
                       *
                       *
                       **
                    80 **
                       **
                       **
                       **
                       **
PERCENTAGE             **
                    60 **
  OF THE               **
                       **
TABLE'S TOTAL          **
                       **
 ENTRIES            40 **
                       **
                       **
                       **
                       **
                    20 **
                       **
                       **
                       **
                       ** ***
                     0 ***************************************************
                       '         '         '         '         '         '
                       0        10        20        30        40        50

                                   TRANSFER TIME IN SECONDS


NOTE. THIS GRAPH IS THE APPROPRIATE SUM OF TABLES "FMPA", "FMPB", AND "FMPC".
```

HISTOGRAM OF TRANSFER TIMES FOR SMALL
FILES (LESS THAN 10 BLOCKS)
BETWEEN THE SPS AND DSC



HISTOGRAM OF TRANSFER TIMES FOR INTERMEDIATE
SIZE FILES (MORE THAN 9 BLOCKS, LESS THAN 100)
BETWEEN THE SPS AND HSC



```
                          HISTOGRAM OF TRANSFER TIMES FOR ALL FILES
                              BETWEEN THE SPS AND DSC

               100 *
                   *
                   *
                   *
                   *
                80 *
                   *
                   **
                   **
  PERCENTAGE       **
                60 **
    OF THE         **
                   **
 TABLE'S TOTAL     **
                   **
   ENTRIES      40 **
                   **
                   **
                   **
                   **
                20 **
                   **
                   **
                   ** *
                   ****         **                   *
                 0 ***************************************************
                   I         I         I         I         I         I
                   0        20        40        60        80       100

                                 TRANSFER TIME IN SECONDS


NOTE, THIS GRAPH IS THE APPROPRIATE SUM OF TABLES, "SPSA", "SPSB", AND "SPSC".
```

```
            UPPER      OBSERVED       PER CENT      CUMULATIVE     CUMULATIVE      MULTIPLE      DEVIATION
            LIMIT      FREQUENCY      OF TOTAL      PERCENTAGE     REMAINDER       OF MEAN       FROM MEAN

              820             15          0.08           99.77           0.23         6.230           4.339
              830              3          0.02           99.79           0.21         6.306           4.402
              840              1          0.01           99.79           0.21         6.382           4.465
              850             40          0.21          100.00           0.00         6.458           4.528
    TABLE DISC
    ENTRIES IN TABLE         MEAN ARGUMENT          STANDARD DEVIATION           SUM OF ARGUMENTS:

              9773              27.442                         29.074             268191.000:

            UPPER      OBSERVED       PER CENT      CUMULATIVE     CUMULATIVE      MULTIPLE      DEVIATION
            LIMIT      FREQUENCY      OF TOTAL      PERCENTAGE     REMAINDER       OF MEAN       FROM MEAN

                3           1094         11.19           11.19          88.81         0.109          -0.840
                6            856          8.76           19.95          80.05         0.219          -0.737
                9            875          8.95           28.91          71.09         0.328          -0.633
               12            831          8.50           37.41          62.59         0.437          -0.530
               15            848          8.68           46.09          53.91         0.547          -0.427
               18            871          8.91           55.00          45.00         0.656          -0.324
               21            834          8.53           63.53          36.47         0.765          -0.221
               24            921          9.42           72.96          27.04         0.875          -0.117
               27             68          0.70           73.65          26.35         0.984          -0.014
               30             75          0.77           74.42          25.58         1.093           0.088
               33            101          1.03           75.45          24.55         1.203           0.191
               36             93          0.95           76.40          23.60         1.312           0.294
               39             83          0.85           77.25          22.75         1.421           0.398
               42             92          0.94           78.20          21.80         1.530           0.501
               45            102          1.04           79.24          20.76         1.640           0.604
               48            104          1.06           80.30          19.70         1.749           0.707
               51            108          1.11           81.41          18.59         1.858           0.810
               54             91          0.93           82.34          17.66         1.968           0.913
               57             95          0.97           83.31          16.69         2.077           1.017
               60             86          0.88           84.19          15.81         2.186           1.120
               63             90          0.92           85.11          14.89         2.296           1.223
               66             81          0.83           85.94          14.06         2.405           1.326
               69             86          0.88           86.82          13.18         2.514           1.429
               72             99          1.01           87.83          12.17         2.624           1.533
               75             91          0.93           88.76          11.24         2.733           1.636
               78             78          0.80           89.56          10.44         2.842           1.739
               81             91          0.93           90.49           9.51         2.952           1.842
               84            102          1.04           91.54           8.46         3.061           1.945
               87             93          0.95           92.49           7.51         3.170           2.049
               90             88          0.90           93.39           6.61         3.280           2.152
               93             99          1.01           94.40           5.60         3.389           2.255
               96             90          0.92           95.32           4.68         3.498           2.358
               99             83          0.85           96.17           3.83         3.608           2.461
              102             79          0.81           96.98           3.02         3.717           2.564
              105             67          0.69           97.67           2.33         3.826           2.668
              108             70          0.72           98.38           1.62         3.936           2.771
              111             61          0.62           99.01           0.99         4.045           2.874
              114             33          0.34           99.35           0.65         4.154           2.977
              117             33          0.34           99.68           0.32         4.264           3.080
              120             24          0.25           99.93           0.07         4.373           3.184
              123              7          0.07          100.00           0.00         4.482           3.287
    TABLE GPHA
```

11-A-131

```
TURNAROUND STATISTICS FOR MESSAGES AND FILES
     TABLE TOTGP
     ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION          SUM OF ARGUMENTS:

                1107                130.061                          122.197            143977.000:

            UPPER       OBSERVED       PER CENT       CUMULATIVE       CUMULATIVE       MULTIPLE       DEVIATION
            LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE       REMAINDER        OF MEAN        FROM MEAN

              100            520          46.97            46.97            53.03          0.769          -0.245
              200            256          23.13            70.10            29.90          1.538           0.572
              300            286          25.84            95.93             4.07          2.307           1.391
              400             26           2.35            98.28             1.72          3.075           2.209
              500              2           0.18            98.46             1.54          3.844           3.027
              600             13           1.17            99.64             0.36          4.613           3.846
              700              1           0.09            99.73             0.27          5.382           4.664
              800              1           0.09            99.82             0.18          6.151           5.482
              900              2           0.18           100.00             0.00          6.920           6.301
     TABLE TOTAL
     ENTRIES IN TABLE          MEAN ARGUMENT          STANDARD DEVIATION          SUM OF ARGUMENTS:

               19189                131.626                          158.653            2525771.000:

            UPPER       OBSERVED       PER CENT       CUMULATIVE       CUMULATIVE       MULTIPLE       DEVIATION
            LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE       REMAINDER        OF MEAN        FROM MEAN

               10           8237          42.93            42.93            57.07          0.076          -0.766
               20             13           0.07            42.99            57.01          0.152          -0.703
               30             20           0.10            43.10            56.90          0.228          -0.640
               40             17           0.09            43.19            56.81          0.304          -0.577
               50             11           0.06            43.24            56.76          0.380          -0.513
               60             14           0.07            43.32            56.68          0.456          -0.450
               70              9           0.05            43.36            56.64          0.532          -0.387
               80             16           0.08            43.45            56.55          0.608          -0.324
               90             14           0.07            43.52            56.48          0.684          -0.261
              100            119           0.62            44.14            55.86          0.760          -0.198
              110           1119           5.83            49.97            50.03          0.836          -0.135
              120           1345           7.01            56.98            43.02          0.912          -0.072
              130           1645           8.57            65.55            34.45          0.988          -0.009
              140            577           3.01            68.56            31.44          1.064           0.053
              150            389           2.03            70.59            29.41          1.140           0.116
              160            159           0.83            71.42            28.58          1.216           0.179
              170             58           0.30            71.72            28.28          1.292           0.242
              180             41           0.21            71.93            28.07          1.368           0.305
              190             31           0.16            72.09            27.91          1.443           0.368
              200             19           0.10            72.19            27.81          1.519           0.431
              210             18           0.09            72.29            27.71          1.595           0.494
              220             17           0.09            72.37            27.63          1.671           0.557
              230             19           0.10            72.47            27.53          1.747           0.620
              240             17           0.09            72.56            27.44          1.823           0.683
              250              5           0.03            72.59            27.41          1.899           0.746
```

11-A-132

```
ARDEN HILLS CYBER 175 GPSS V/6000                CRH GPSS V/6000   VER. 1.2 PSR 453     03/17/79  15.03.54.   PAGE     361
       UPPER       OBSERVED          PER CENT       CUMULATIVE      CUMULATIVE      MULTIPLE       DEVIATION
       LIMIT       FREQUENCY         OF TOTAL       PERCENTAGE      REMAINDER       OF MEAN        FROM MEAN

         260             432             2.25            74.84          25.16          1.975           0.809
         270             867             4.52            79.36          20.64          2.051           0.872
         280            1015             5.29            84.65          15.35          2.127           0.935
         290             764             3.98            88.63          11.37          2.203           0.998
         300             450             2.35            90.97           9.03          2.279           1.061
         310             243             1.27            92.24           7.76          2.355           1.124
         320             103             0.54            92.78           7.22          2.431           1.187
         330              65             0.34            93.12           6.88          2.507           1.250
         340              43             0.22            93.34           6.66          2.583           1.313
         350              25             0.13            93.47           6.53          2.659           1.376
         360              22             0.11            93.58           6.42          2.735           1.439
         370              27             0.14            93.73           6.27          2.811           1.502
         380              18             0.09            93.82           6.18          2.887           1.566
         390               8             0.04            93.86           6.14          2.963           1.629
         400               5             0.03            93.89           6.11          3.039           1.692
         410               4             0.02            93.91           6.09          3.115           1.755
         420               2             0.01            93.92           6.08          3.191           1.818
         430               3             0.02            93.93           6.07          3.267           1.881
         440               3             0.02            93.95           6.05          3.343           1.944
         450               6             0.03            93.98           6.02          3.419           2.007
         460              13             0.07            94.05           5.95          3.495           2.070
         470              14             0.07            94.12           5.88          3.571           2.133
         480              26             0.14            94.26           5.74          3.647           2.196
         490              23             0.12            94.38           5.62          3.723           2.259
         500              24             0.13            94.50           5.50          3.799           2.322
         510              41             0.21            94.72           5.28          3.875           2.385
         520              51             0.27            94.98           5.02          3.951           2.448
         530              75             0.39            95.37           4.63          4.027           2.511
         540             103             0.54            95.91           4.09          4.103           2.574
         550             129             0.67            96.58           3.42          4.179           2.637
         560             142             0.74            97.32           2.68          4.254           2.700
         570             128             0.67            97.99           2.01          4.330           2.763
         580             108             0.56            98.55           1.45          4.406           2.826
         590              72             0.38            98.93           1.07          4.482           2.889
         600              34             0.18            99.10           0.90          4.558           2.952
         610              28             0.15            99.25           0.75          4.634           3.015
         620              15             0.08            99.33           0.67          4.710           3.078
         630               8             0.04            99.37           0.63          4.786           3.141
         640               6             0.03            99.40           0.60          4.862           3.204
         650               7             0.04            99.44           0.56          4.938           3.267
         660               3             0.02            99.45           0.55          5.014           3.330
         670               5             0.03            99.48           0.52          5.090           3.393
         680               3             0.02            99.49           0.51          5.166           3.456
         690               3             0.02            99.51           0.49          5.242           3.519
         700               1             0.01            99.52           0.48          5.318           3.582
         710               1             0.01            99.52           0.48          5.394           3.646
         720               3             0.02            99.54           0.46          5.470           3.709
         730               1             0.01            99.54           0.46          5.546           3.772
         740               1             0.01            99.55           0.45          5.622           3.835
         750               4             0.02            99.57           0.43          5.698           3.898
         760               1             0.01            99.57           0.43          5.774           3.961
         770               3             0.02            99.59           0.41          5.850           4.024
         780               5             0.03            99.61           0.39          5.926           4.087
         790               6             0.03            99.65           0.35          6.002           4.150
         800               5             0.03            99.67           0.33          6.078           4.213
         810               4             0.02            99.69           0.31          6.154           4.276
```

```
ARDEN HILLS CYBER 175 GPSS V/6000              CRM GPSS V/6000   VER. 1.2 PSR 453      03/17/79  15.03.54.    PAGE     363
   ENTRIES IN TABLE         MEAN ARGUMENT            STANDARD DEVIATION          SUM OF ARGUMENTS:

            63                   1.460                            0.895              92.000:

          UPPER      OBSERVED        PER CENT       CUMULATIVE      CUMULATIVE       MULTIPLE        DEVIATION
          LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE      REMAINDER        OF MEAN         FROM MEAN

              1             44          69.84            69.84           30.16          0.685           -0.513
              2             13          20.63            90.48            9.52          1.370            0.603
              3              4           6.35            96.83            3.17          2.054            1.721
              4              1           1.59            98.41            1.59          2.739            2.838
              5              0           0.00            98.41            1.59          3.424            3.955
              6              1           1.59           100.00            0.00          4.109            5.073
   TABLE FILE3
   ENTRIES IN TABLE         MEAN ARGUMENT            STANDARD DEVIATION          SUM OF ARGUMENTS:

            63                   1.460                            0.895              92.000:

          UPPER      OBSERVED        PER CENT       CUMULATIVE      CUMULATIVE       MULTIPLE        DEVIATION
          LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE      REMAINDER        OF MEAN         FROM MEAN

              5             62          98.41            98.41            1.59          3.424            3.955
             10              1           1.59           100.00            0.00          6.848            9.543
   TABLE FMPA
   ENTRIES IN TABLE         MEAN ARGUMENT            STANDARD DEVIATION          SUM OF ARGUMENTS:

           205                 280.756                          150.624           57555.000:

          UPPER      OBSERVED        PER CENT       CUMULATIVE      CUMULATIVE       MULTIPLE        DEVIATION
          LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE      REMAINDER        OF MEAN         FROM MEAN

            250            144          70.24            70.24           29.76          0.890           -0.203
            500             33          16.10            86.34           13.66          1.781            1.456
            750             27          13.17            99.51            0.49          2.671            3.115
           1000              1           0.49           100.00            0.00          3.562            4.775
   TABLE FMPB
   ENTRIES IN TABLE         MEAN ARGUMENT            STANDARD DEVIATION          SUM OF ARGUMENTS:

            46                2334.174                         1823.707          107372.000:

          UPPER      OBSERVED        PER CENT       CUMULATIVE      CUMULATIVE       MULTIPLE        DEVIATION
          LIMIT      FREQUENCY       OF TOTAL       PERCENTAGE      REMAINDER        OF MEAN         FROM MEAN

            500              3           6.52             6.52           93.48          0.214           -1.005
           1000             17          36.96            43.48           56.52          0.428           -0.731
           1500              2           4.35            47.83           52.17          0.643           -0.456
           2000              2           4.35            52.17           47.83          0.857           -0.182
           2500              2           4.35            56.52           43.48          1.071            0.091
           3000              0           0.00            56.52           43.48          1.285            0.365
           3500             11          23.91            80.43           19.57          1.499            0.639
           4000              1           2.17            82.61           17.39          1.714            0.913
           4500              2           4.35            86.96           13.04          1.928            1.188
           5000              1           2.17            89.13           10.87          2.142            1.462
           5500              1           2.17            91.30            8.70          2.356            1.736
           6000              1           2.17            93.48            6.52          2.571            2.010
           6500              2           4.35            97.83            2.17          2.785            2.284
           7000              1           2.17           100.00            0.00          2.999            2.558
   TABLE FMPC
```

```
ARDEN HILLS CYBER 175 GPSS V/6000                  CRM GPSS V/6000   VER. 1.2 PSR 453      03/17/79  15.03.54.   PAGE    364
     ENTRIES IN TABLE          MEAN ARGUMENT             STANDARD DEVIATION          SUM OF ARGUMENTS:

                   16                  6.687                          3.049               107.000:

             UPPER        OBSERVED       PER CENT       CUMULATIVE      CUMULATIVE      MULTIPLE       DEVIATION
             LIMIT        FREQUENCY      OF TOTAL       PERCENTAGE      REMAINDER       OF MEAN        FROM MEAN

                 5               8          50.00            50.00           50.00         0.748          -0.552
                10               6          37.50            87.50           12.50         1.495           1.086
                15               2          12.50           100.00            0.00         2.243           2.726
     TABLE FILE1
     ENTRIES IN TABLE          MEAN ARGUMENT             STANDARD DEVIATION          SUM OF ARGUMENTS:

                  267                  0.715                          1.981               191.000:

             UPPER        OBSERVED       PER CENT       CUMULATIVE      CUMULATIVE      MULTIPLE       DEVIATION
             LIMIT        FREQUENCY      OF TOTAL       PERCENTAGE      REMAINDER       OF MEAN        FROM MEAN

                 1             229          85.77            85.77           14.23         1.398           0.144
                 2               2           0.75            86.52           13.48         2.796           0.648
                 3              12           4.49            91.01            8.99         4.194           1.153
                 4               7           2.62            93.63            6.37         5.592           1.658
                 5               6           2.25            95.88            4.12         6.990           2.162
                 6               5           1.87            97.75            2.25         8.387           2.667
                 7               2           0.75            98.50            1.50         9.785           3.172
                 8               0           0.00            98.50            1.50        11.183           3.676
                 9               1           0.37            98.88            1.12        12.581           4.181
                10               1           0.37            99.25            0.75        13.979           4.686
                11               0           0.00            99.25            0.75        15.377           5.190
                12               1           0.37            99.63            0.37        16.775           5.695
                13               0           0.00            99.63            0.37        18.173           6.200
                14               1           0.37           100.00            0.00        19.571           6.704
     TABLE SPSA
     ENTRIES IN TABLE          MEAN ARGUMENT             STANDARD DEVIATION          SUM OF ARGUMENTS:

                  142                988.993                        767.398            140437.000:

             UPPER        OBSERVED       PER CENT       CUMULATIVE      CUMULATIVE      MULTIPLE       DEVIATION
             LIMIT        FREQUENCY      OF TOTAL       PERCENTAGE      REMAINDER       OF MEAN        FROM MEAN

               500              48          33.80            33.80           66.20         0.506          -0.636
              1000              43          30.28            64.08           35.92         1.011           0.014
              1500              26          18.31            82.39           17.61         1.517           0.666
              2000               9           6.34            88.73           11.27         2.022           1.317
              2500              10           7.04            95.77            4.23         2.528           1.969
              3000               3           2.11            97.89            2.11         3.033           2.621
              3500               1           0.70            98.59            1.41         3.539           3.272
              4000               0           0.00            98.59            1.41         4.045           3.924
              4500               2           1.41           100.00            0.00         4.550           4.575
     TABLE SPSB
     ENTRIES IN TABLE          MEAN ARGUMENT             STANDARD DEVIATION          SUM OF ARGUMENTS:

                   46                 18.913                         15.829               870.000:

             UPPER        OBSERVED       PER CENT       CUMULATIVE      CUMULATIVE      MULTIPLE       DEVIATION
             LIMIT        FREQUENCY      OF TOTAL       PERCENTAGE      REMAINDER       OF MEAN        FROM MEAN

                 1               0           0.00             0.00          100.00         0.053          -1.131
                 2               0           0.00             0.00          100.00         0.106          -1.067
```

```
ARDEN HILLS CYBER 175 GPSS V/6000               CRM GPSS V/6000    VER. 1.2 PSR 453       03/17/79  15.03.54.    PAGE      365
             UPPER        OBSERVED        PER CENT      CUMULATIVE      CUMULATIVE     MULTIPLE      DEVIATION
             LIMIT       FREQUENCY        OF TOTAL      PERCENTAGE      REMAINDER      OF MEAN       FROM MEAN

                 3               2            4.35            4.35           95.65        0.159         -1.004
                 4               0            0.00            4.35           95.65        0.211         -0.941
                 5              16           34.78           39.13           60.87        0.264         -0.878
                 6               2            4.35           43.48           56.52        0.317         -0.815
                 7               0            0.00           43.48           56.52        0.370         -0.752
                 8               1            2.17           45.65           54.35        0.423         -0.688
                 9               0            0.00           45.65           54.35        0.476         -0.625
                10               0            0.00           45.65           54.35        0.529         -0.562
                11               1            2.17           47.83           52.17        0.582         -0.499
                12               0            0.00           47.83           52.17        0.634         -0.436
                13               0            0.00           47.83           52.17        0.687         -0.373
                14               3            6.52           54.35           45.65        0.740         -0.309
                15               0            0.00           54.35           45.65        0.793         -0.246
                16               0            0.00           54.35           45.65        0.846         -0.183
                17               0            0.00           54.35           45.65        0.899         -0.120
                18               0            0.00           54.35           45.65        0.952         -0.057
                19               0            0.00           54.35           45.65        1.005          0.005
                20               0            0.00           54.35           45.65        1.057          0.069
                21               1            2.17           56.52           43.48        1.110          0.132
                22               0            0.00           56.52           43.48        1.163          0.195
                23               0            0.00           56.52           43.48        1.216          0.258
                24               0            0.00           56.52           43.48        1.269          0.321
                25               0            0.00           56.52           43.48        1.322          0.385
                26               6           13.04           69.57           30.43        1.375          0.448
                27               2            4.35           73.91           26.09        1.428          0.511
                28               3            6.52           80.43           19.57        1.480          0.574
                29               0            0.00           80.43           19.57        1.533          0.637
                30               1            2.17           82.61           17.39        1.586          0.700
                31               1            2.17           84.78           15.22        1.639          0.764
                32               0            0.00           84.78           15.22        1.692          0.827
                33               0            0.00           84.78           15.22        1.745          0.890
                34               0            0.00           84.78           15.22        1.798          0.953
                35               0            0.00           84.78           15.22        1.851          1.016
                36               0            0.00           84.78           15.22        1.903          1.079
                37               0            0.00           84.78           15.22        1.956          1.143
                38               0            0.00           84.78           15.22        2.009          1.206
                39               0            0.00           84.78           15.22        2.062          1.269
                40               2            4.35           89.13           10.87        2.115          1.332
                41               0            0.00           89.13           10.87        2.168          1.395
                42               0            0.00           89.13           10.87        2.221          1.458
                43               0            0.00           89.13           10.87        2.274          1.522
                44               1            2.17           91.30            8.70        2.326          1.585
                45               0            0.00           91.30            8.70        2.379          1.648
                46               0            0.00           91.30            8.70        2.432          1.711
                47               0            0.00           91.30            8.70        2.485          1.774
                48               0            0.00           91.30            8.70        2.538          1.838
                49               1            2.17           93.48            6.52        2.591          1.901
                50               3            6.52          100.00            0.00        2.644          1.964
```

```
                 FULLWORD   SAVEVALUES


NUMBER ........ CONTENTS   NUMBER ........ CONTENTS   NUMBER ........ CONTENTS   NUMBER ........ CONTENTS   NUMBER ........ CONTENTS
    17               63        18                7    COMNT                9        22               45        23             7671
    24             7671        25               60       26                2    CYLMN             1000    CYLAV             5500
CYLDV              4500    ROTMN                 1    ROTAV             1251    ROTDV             1250    BLOK1            11500
ROTMX              2501    TRKRT                65    CLOCK         48943742    FMPRT               65    BFFMP                3
SPSRT               273    BFSPS                 3    DSCRT              156    BFDSC                3    END         1000000000
BLOK0             27000    DBLK0              1500    DBLK1             1500    COUNT                1    CARD              1251
DSCN                  1    DRIVE                 1    PRIOR               62    SLOP               200    IOPEC               10
LAST                  2    BLOK2             42000    BLOK3           108000    DBLK2            10000    DBLK3            10000
IOPGR                30    GPHRT              1638    GPHWT             5000    DWAIT              500    BLK2A            10500
BLK3A             27000    DBK2A              2500    DBK3A             2500    ILSPS               20    ILGPH                5
GPHSZ                16    COPYS                 4    BSIZE               64


                  THROUGHPUTS BY JOB SIZE.

             FOR JOBS WITH FMP TIME LESS THAN OR EQUAL TO 120 SEC
             THROUGHPUT IS       29.44   PER HOUR

             FOR JOBS WITH FMP TIME BETWEEN 120 SEC AND 20 MIN
             THROUGHPUT IS        3.27   PER HOUR

             FOR JOBS WITH FMP TIME BETWEEN 20 MIN AND ONE HOUR
             THROUGHPUT IS        0.00   PER HOUR

             FOR JOBS WITH FMP TIME MORE THAN ONE HOUR
             THROUGHPUT IS        0.00   PER HOUR
```

REFERENCES

1. NASA CR-152059, Preliminary Study for a Numerical Aerodynamic Simulation Faclity - Final Report. Oct. 1977.

2. NASA CR-152108, Preliminary Study for a Numerical Aerodynamic Simulation Facility - Final Report - Phase 1 Extension. Feb. 1978.

3. Machenhauer, B.; and Rasmussen, E.: On the Integration of the Spectral Hydrodynamical Equations by a Transform Method. Institut for Teoretisk Meteorologi, Kobenhavns Universitet, Copenhagen, 1972.

4. Hung, C. M.; and MacCormack, R. W.: Numerical Solution of Supersonic Laminar Flow Over a Three-Dimensional Compression Corner. AIAA 10th Fluids and Plasmadynamics Conference, June 1977.

5. Lambiotte, J. S., Jr.; and Voigt, R. G.: The Solution of Tridiagonal Linear Systems on the CDC STAR-100 Computer. ACM Transactions on Mathematical Software, Dec. 1978.

6. Soll, D. B.; Habra, N. R.; and Russell, G. L.: Experience with a Vectorized General Circulation Climate Model on STAR-100. High Speed Computer and Algorithm Organization, Academic Press, Inc. Nov. 1977, pp. 311-312.

7. Lorenz, E. N.: Energy and Numerical Weather Prediction. Tellus, vol. 12, no. 4, Nov. 1960, pp. 364-373.

8. Peng, L.: A Simple Numerical Experiment Concerning the General Circulation in the Low Stratosphere. Pure and Applied Geophysics, vol. 61, no.2, May 1965, pp. 191-218.

9. Clark, J. H. E.: A Quasi-Geostrophic Model of the Winter Stratospheric Circulation. Monthly Weather Review, vol. 98, no. 6, June 1970, pp. 443-461.

10. Lindzen, R. S.; and Goody, R.: Radiative and Photo-Chemical Processes in Mesopheric Dynamics: Part 1, Models for Radiative and Photochemical Processes. Journal of the Atmospheric Science, vol. 22, no. 4, July 1965, pp. 341-348.

11. McConnell, J. C.; and McElroy, M. B.: Odd Nitrogen in the Atmosphere. Journal of the Atmospheric Science, vol. 30, no. 8, Nov. 1973, pp. 1465-1480.

12. Lorenz, E. N.: An N-Cycle Time-Differencing Scheme for Stepwise Numerical Integration. Monthly Weather Review, vol. 99, no. 8, Aug. 1971, pp. 644-648. `trailer